Pedro Bueno Mota

Licenciado em Ciências de Engenharia
Electrotécnica e de Computadores

Game Wizard

Dissertação para obtenção do Grau de Mestre em
Engenharia Electrotécnica e de Computadores

Orientador: Tiago Oliveira Machado de Figueiredo Cardoso
Professor Auxiliar, FCT - UNL

Co-Orientador: Yves Rybarczyk
Professor Auxiliar, FCT - UNL

Júri:

Presidente: Doutor Pedro Miguel Ribeiro Pereira, FCT - UNL

Arguente: Doutor Vítor Manuel Pereira Duarte dos Santos, ISEGI - UNL

Vogal: Doutor Tiago Oliveira Machado de Figueiredo Cardoso, FCT - UNL

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Setembro, 2014

Faculdade de Ciências e Tecnologia

Universidade Nova de Lisboa

Game Wizard

Pedro Bueno Mota

Setembro, 2014

Game Wizard

Ao meu Pai, à minha Mãe e à minha namorada.

Agradecimentos

Ao terminar esta dissertação ponho fim a uma etapa muito significativa da minha vida e, sem dúvida, a mais importante da minha vida académica até ao momento. Gostaria de começar por agradecer ao Professor Tiago Cardoso, por todo o apoio e orientação prestada ao longo deste trabalho, à Faculdade de Ciências e Tecnológica da Universidade Nova de Lisboa por me acolher e por todas as boas condições oferecidas para a minha formação.

Ao meu Pai e à minha Mãe, obrigado por tudo. Por sempre me apoiarem e incentivarem em todos os momentos difíceis e por nunca deixarem de acreditar em mim. À Sílvia e ao José por nunca me deixarem estudar de barriga vazia. À minha avó, ao meu tio e aos meus primos, pelo carinho e confiança incondicional que sempre demonstraram.

Agradeço a todo o meu grupo de amigos, que estiveram e estão sempre presentes, um obrigado a vocês Bruno Caixinha, Bruno Fernandes, David Aleixo, Diogo Pinto, Hugo Pereira, João Eusébio, Manuel Ferro, Nuno Pinheiro, Nuno Barreira, Pedro Oliveira, Raquel Melo, Telma Canário e Tiago Pereira. Por fim, um obrigado especial à minha namorada Ana por todo o apoio, carinho e paciência incondicional que sempre demonstrou.

Muito Obrigado.

Resumo

Hoje em dia, há um aumento evidente dos produtos ou soluções sob medida, com o objectivo de melhor se adaptarem às necessidades dos perfis de clientes. Num outro contexto, existe um largo crescimento da presença da computação informática nas salas de aulas, de maneira a auxiliar o professor a cativar e motivar os alunos, através de jogos educativos. Contudo, nem todos os docentes têm conhecimentos informáticos avançados, então não têm forma de personalizar jogos, o que beneficiaria as crianças com soluções adaptadas às suas necessidades. Com isto, foi idealizado um sistema, que assenta sobre uma *framework* de criação de jogos, e que possibilite a um utilizador com menos habilidades informáticas a possibilidade de criar um jogo. Apesar de existirem diversos jogos desse tipo, seria vantajoso um professor poder criar um jogo, personalizando-o de acordo com as necessidades do aluno e com as competências que se desejam desenvolver. Este sistema consiste numa interface em que o utilizador segue uma série de passos definidos, do qual resulta um ficheiro com todas as características de jogo, que será depois importado para uma *framework*, permitindo assim a criação de diferentes jogos, desde que se enquadrem na mesma mecânica.

Palavras-chave: *Wizard,* Motor de Jogo, Jogos Educativos

Abstract

Nowadays there is a clear increase of products or tailored solutions with the goal of better adept to the needs of clients. In another context there is a wide growth of informatics computation presence in the class rooms, in order to assist the teacher to captivate and motivate the students using educational games. However not all teachers have the advanced computer knowledge that would allow them to personalize games, which would benefit children by providing tailored solutions to their specific needs. Having this problem into consideration it was designed/idealized a system that relies on a framework of game creation that allows a user with less informatics skills to create a game based on a model. Although there are many games of this type it would be useful and innovative for a teacher to be able to create a game that is personalized to the students' needs and according to the skills that they wish to develop. This system consists in an interface where the user has to follow some established steps, in order to achieve a file containing all the game features, which is later imported to a framework. That system allows the creation of different games, as long as these games come under the same game mechanics.

Keywords: Wizard, Game Engine, Educational Game

Índice de Conteúdo

Lista de Figuras

Lista de Tabelas

Acrónimos

2D	Espaço Bidimensional
3D	Espaço Tridimensional
DLL	Dynamic-link Library
FPS	First Person Shooter
IDE	Integrated Development Environment
iOS	Sistema Operativo Apple
GML	Game Maker Language
PS	PlayStation
XML	eXtensible Markup Language

1 Introdução

Actualmente, grande parte das nossas acções e relações são mediadas por objectos electrónicos. O desenvolvimento da indústria nos séculos XVII e XVIII marca o início das transformações que, nos tempos que correm, ganham uma sofisticação técnica inimaginável naquele período. A automatização e a robótica, através de mecanismos considerados inteligentes, tornaram algumas máquinas capazes de executar diferentes funções e de realizarem complexas tarefas a partir de uma acção humana.

A presença dos computadores na nossa sociedade é bastante significativa e continua a aumentar a cada dia que passa. Este ritmo imposto pelo desenvolvimento tecnológico levou a uma alteração no uso dos sentidos, exigindo outros movimentos físicos, de gestos, de linguagem. Esse processo atinge os adultos, e principalmente as crianças, que são integradas nessa aventura tecnológica via jogos electrónicos.

Os jogos representam o que há de mais moderno e inovador em termos de diversão electrónica e não são da exclusiva utilização do sector mais jovem, pois também os adultos vêem nos jogos um escape aos problemas da vida real respondendo assim à necessidade básica de relaxamento. “Estes oferecem às pessoas um meio de diversão sem precedentes, independentemente da sua idade ou género” (Cabral, 1997).

Um jogo é um sistema em que os utilizadores encontram um conflito artificial, definido por regras e do qual se obtém um resultado quantificável. Estes devem ser agradáveis de se utilizar, uma vez que o seu principal objectivo é proporcionar entretenimento para as pessoas. Um jogo electrónico é assim um jogo comum, em que o utilizador interage com um dispositivo e o seu resultado é exibido através de um ecrã, que mostra as imagens em movimento.

“Em resumo, eles podem ser descritos como a interacção entre o jogador e as imagens que aparecem numa tela, mediada por um processador e uma interface física” (Goto, 2005). Esta interacção entre máquina e utilizador é normalmente feita através da visão e da audição, porém são também bastante utilizados dispositivos de vibração ou sensores de movimento (Flausino, 2006).

A criação de jogos electrónicos é um grande desafio, pois trata-se de uma tarefa multidisciplinar em que são necessários conhecimentos e habilidades de várias áreas técnicas, como é

possível observar através da Figura 1.1. A existência de uma grande equipa, com elementos especializados em diferentes áreas é necessária para a criação de jogos electrónicos, fazendo com que a indústria dos jogos electrónicos seja uma das maiores do mundo, sendo mesmo maior que a indústria do cinema, continuando em constante expansão (Flausino, 2006).

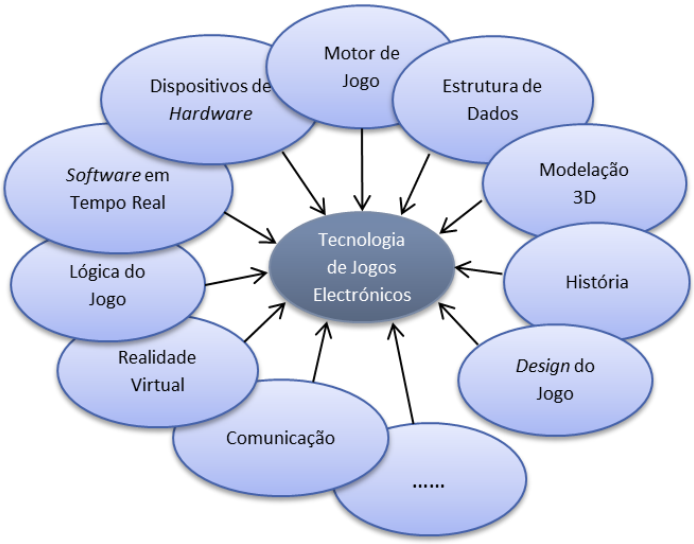

Figura 1.1 - Tecnologia de Jogos Electrónicos (Cardoso, 2013)

Para desenvolver um jogo virtual uma equipa deve ser composta, pelo menos, por cinco elementos diferentes:

- ***Game Designer:*** Desempenha um papel de liderança, definindo o estilo e assegurando a consistência de como o jogo se comporta. É também o responsável pela criação da história que servirá de base ao jogo;

- **Programador:** O programador é o responsável de criar os *scripts* que correm e controlam o jogo, incorporando a visão do *game designer* e interligando os aspectos artísticos com os aspectos sonoros num jogo;

- **Artista Gráfico:** É nele que recai a tarefa de criar os elementos gráficos do jogo, desde imagens de fundo até aos personagens, assim como de material de *marketing*. Tem a particularidade de estar em contacto constante com o programador, de forma serem garantidos os requisitos técnicos;

- **Artista de Som:** Muitas vezes menosprezado, o som é de vital importância para criar atmosfera no jogo. Tal como o artista gráfico, tem que trabalhar em conjunto com o programador, que tem que construir a infra-estrutura para o som.

- **Produtor:** O produtor de uma equipa actua como um gestor de projectos, garantindo que as contribuições das diferentes áreas sejam capazes de completar um jogo com qualidade, dentro do orçamento e no tempo destinado. Este cria e gere a calendarização, assegurando a disponibilidade dos recursos e guiando a equipa ao sucesso.

Para empresas com maior capacidade financeira, que criam jogos de enorme complexidade, existe um maior número de pessoas dedicadas às mesmas áreas, ou subáreas das acima mencionadas, devido à dimensão dos seus projectos. Assim, dentro da equipa de um jogo, existem outras subequipas destinadas a executar tarefas específicas, por exemplo, ao invés de existir um programador, existe uma equipa com membros destinados a tarefas de programação com foco em diferentes aspectos, assim como podem existir diversos artistas gráficos, cada um com a sua função dedicada a um elemento específico.

Com o crescimento em massa dos jogos destinados a dispositivos móveis não é surpresa que tenha igualmente crescido o número de programadores de jogos individuais ou de pequenos grupos de pessoas, designados como *indie developers* que, como é comprovado pela sondagem feita na *Game Developers Conference* 2013[1], em que 53% dos estúdios Norte Americanos são classificados como *indie*. Estes programadores individuais não têm os meios financeiros nem os recursos de uma grande empresa de videojogos. Como alternativa utilizam ferramentas que lhes permitam implementar as suas ideias e alcançarem o resultado desejado sem ser necessário o domínio de todas as áreas presentes na criação de um jogo virtual.

Com uma boa ideia para um jogo, algum conhecimento sobre todas as áreas e com poderosas ferramentas à sua disposição um indivíduo consegue obter um volume de vendas superior a alguns jogos criados por grandes empresas.

Os programadores *indie* encontraram um meio de conseguirem produzir os seus jogos criando-os por partes, utilizando ferramentas diferentes para os vários elementos do jogo e inovando no método de desenvolvimento dos videojogos.

Hoje em dia, grande parte dos jogos são construídos de uma forma modular através de instrumentos que criam os jogos por cima de um *game engine*. Estes *game engines* não estão directamente interligados com o comportamento dos elementos do jogo ou com o seu ambiente. Estes motores de jogo incluem os módulos responsáveis por *inputs*, *outputs* e físicas genéricas para o mundo do jogo e possibilitam o desenvolvimento de inúmeros modelos de jogos, ou seja, são programas de computador com um conjunto de bibliotecas de um modo abstracto, responsáveis pela harmonia entre os vários recursos dos jogos (Gregory, 2009).

[1] Maior encontro anual de programadores de videojogos profissionais, focado na aprendizagem, inspiração e comunicação.

1.1 Contexto e Motivação

A particularização de soluções está a tornar-se, rapidamente, um factor importante na educação, na saúde e no marketing. Ainda que a personalização de experiências interactivas esteja no seu início, a capacidade de serem criadas experiências únicas, que tiram vantagem das diferenças e das capacidades pessoais de cada indivíduo, tais como gostos, preferências e limitações, é praticamente obrigatória (Riedl, 2010).

Associado a este facto, também a introdução do computador na escola é justificada pelo aumento motivacional que este cria nos alunos, pois estes sentem-se mais estimulados e focados pelas oportunidades de resolução de problemas especiais e distintos que são oferecidas pelos computadores. Estes problemas são conseguidos através de jogos educativos, que devem proporcionar um ambiente crítico, em que o aluno consiga alcançar os seus objectivos, melhorando o seu raciocínio ou a sua capacidade motora através da resolução de problemas agradáveis (Moratori, 2003).

Este trabalho pretende contribuir para esta área específica, facultando a capacidade de um utilizador comum de computador criar um jogo personalizado para alvos específicos, sem ser necessário o domínio de ferramentas especializadas para esse fim ou de qualquer tipo de conhecimento avançado na implementação de jogos electrónicos. Este resultado será alcançado com uma aplicação integrada sobre um motor de jogo em que o utilizador segue uma série de passos previamente definidos para a sua criação. Este sistema será definido de *Wizard*, que substituirá uma equipa de programação, no processo de transformação do conceito de um jogo para o seu trabalho final.

Este projecto será aplicado ao jogo **Reino dos Fonemas**, desenvolvido pelo **DIFERENÇAS – Centro de Desenvolvimento Infantil** (Diferenças) - tornando-se uma vantagem para um centro com estas características ter a capacidade de desenvolver jogos diferentes, destinados a jogadores realmente distintos. A parceria com o centro Diferenças e a sua utilização desta proposta, servirá como validação do projecto.

1.2 Metodologia de Trabalho

Para esta dissertação foi considerada a metodologia de trabalho clássico. Este método é constituído por sete passos. Começando por uma contextualização teórica, passando progressivamente para uma visão mais prática do sistema e terminando com uma prova de conceito e uma análise aos resultados obtidos. Como é uma metodologia iterativa, caso os resultados não sejam os previstos, é de grande interesse voltar ao primeiro passo e tentar uma nova abordagem.

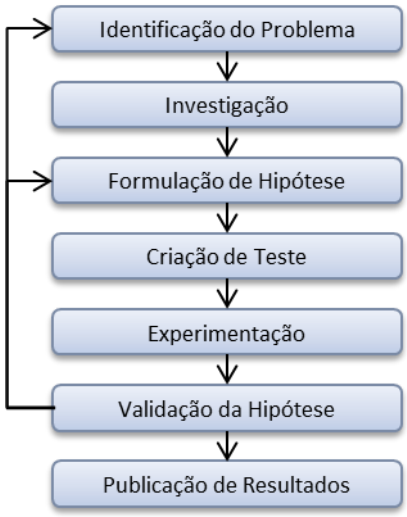

Figura 1.2 - Fases da metodologia de pesquisa clássica (Schafersman, 1994)

Este método é então baseado nos seguintes passos:

1. **Caracterização do problema:** Este é o passo mais importante para a pesquisa. Tem como propósito identificar qual a área de interesse para esta investigação. A caracterização do problema deve ser claramente definida, para que o estudo seja viável. O obstáculo identificado nesta tese é a criação de uma ferramenta que, de uma forma simplificada, possibilite a implementação de jogos.
2. **Investigação:** É o passo no qual a informação relativa à caracterização do problema identificado é adquirida. É então necessário analisar projectos literários e científicos, da área de interesse, de outros autores. Neste caso específico é importante recolher informação sobre jogos educativos e de motores de jogo que permitam a criação de um jogo.
3. **Formulação de hipótese:** A hipótese é formulada baseada na investigação feita de forma a tornar a caracterização do problema mais fácil de perceber e de forma a gerir previsões para o trabalho de pesquisa. A hipótese ajuda a clarificar, a especificar e a focar para o problema de pesquisa e definir os resultados desejáveis. Nesta dissertação, é necessário que a hipótese seja focada no problema e na complexidade que existe na criação de jogos.
4. **Criação de um teste:** Este passo consiste na criação de uma experiência de maneira a validar a hipótese ou rectificar eventuais problemas. Esta experiência é realizada tendo em conta a caracterização do problema e a hipótese formulada.

5. **Teste da hipótese através de experimentação:** Neste passo são definidos diversos testes de forma a tentar prever o resultado, sendo este analisado e interpretado consoante as características do problema**,** para validar a hipótese**.**
6. **Interpretação / Análise de resultados:** Após completado o teste serão analisados os resultados e será verificada a validação da hipótese. Dois desfechos são possíveis. Os resultados poderão ser satisfatórios, indicando que as previsões feitas anteriormente seriam exactas e respondendo à questão do problema (primeiro passo), ou os resultados poderão não ser os esperados, tendo então de se aperfeiçoar a abordagem inicial, ou seja voltar ao passo três e reformular a hipótese.
7. **Conclusões:** A publicação dos resultados é feita de acordo com o resultado da experimentação na área de investigação. Esta etapa só ocorre se o resultado responder à questão definida e deve ser finalizado com uma dissertação sobre a hipótese, assim como este documento.

1.3 Questão Principal

Haverá solução para que uma pessoa sem conhecimentos de programação consiga criar um jogo, tendo um modelo como base?

1.4 Hipótese

Com base na pesquisa das características de vários motores de jogo existentes e com o conhecimento da dificuldade existente na criação de um jogo é formulada a hipótese de uma aplicação, um ***Wizard***, que apoiado na utilização de um *game engine*, facilita a concepção de um jogo sem haver necessidade de conhecimentos prévios de programação por parte do utilizador.

1.5 Esquema da Dissertação

Esta dissertação está dividida em cinco capítulos:

- **Introdução:** No primeiro capítulo é onde se encontram as ideias principais e onde é introduzido o contexto que conduz ao desenvolvimento deste trabalho. É também apresentada a metodologia de trabalho utilizada e a hipótese formulada.

- **Estado da Arte:** Esta secção apresenta elementos de trabalhos relacionados com os que são estudados na área desta tese, proporcionando uma tecnologia que possibilite a implementação do conceito de criação de jogos digitais

- **Proposta:** O terceiro capítulo apresenta a solução criada para validação da hipótese formulada. Começa com a descrição de um modelo de jogo que servirá como teste à hipótese do *Wizard*, seguido de um levantamento de requisitos e especificações que o sistema deverá ter, sendo depois apresentada a solução. Finalmente é apresentado um diagrama de sequência que possibilita uma visão global do comportamento da solução.

- **Validação da Proposta:** Esta secção do documento ostenta os testes efectuados para validar a hipótese formulada. Inicia com a apresentação da metodologia adoptada para testar a hipótese. É então efectuada a demonstração de um cenário de utilização, seguido pela descrição da prova de conceito implementada, e por fim, são apresentados os resultados dos testes efectuados.

Conclusão: O capítulo final, conclusões, sumariza esta dissertação, evidenciando os aspectos mais importantes deste trabalho, assim como um potencial encaminhamento para pesquisas futuras

2 Estado da Arte

Neste capítulo é feita uma extensiva pesquisa de modo a avaliar o estado actual dos jogos electrónicos, mais especificamente, dos jogos que contenham alguma componente educativa, de modo a identificar a vantagem da presença de um *wizard* no âmbito da criação de jogos para a área da educação.

São também identificadas as ferramentas de desenvolvimento de jogos existentes, que por norma têm os respectivos motores de jogo agregados, que possam servir como base à criação automática de jogos.

Foram identificadas várias soluções mas só quatro serão focadas, pois são as que oferecem uma abordagem mais adequada para a solução do problema.

2.1 Jogos Electrónicos

> *"Os jogadores assumem papéis realistas, enfrentam problemas, formulam estratégias, tomam decisões, e obtêm rápida informação sobre as consequências das suas acções." (Abt, 1974)*

O entretenimento imersivo e interactivo, ou jogos electrónicos, emergiu como sendo um capital meio de entretenimento e educacional, exercendo uma enorme influência económica, cultural e social. A sua indústria é equiparada com a de outros *media* em dimensão e popularidade, estabelecendo com os consumidores um meio de entretenimento interactivo. Eles representam como sistemas reais ou imaginários funcionam, convidando os jogadores a interagir com esses sistemas e formularem apreciações sobre eles. (Newman, 2013).

O conceito de jogo electrónico é actualmente considerado um conceito ambíguo pela sua abrangência, estando fortemente ligado aos termos de "videojogo" e "jogo de computador". Usualmente os videojogos aparecem associados a consolas onde se jogam "jogos de vídeo", enquanto os "jogos de computador" representam uma categoria de videojogos que só podem ser jogados numa plataforma específica, no PC. Pelo estado actual dos jogos, e pela distribui-

ção multiplataforma dos jogos oferecida pela indústria, categorizam-se os videojogos e os jogos de computador como jogos electrónicos, pois em ambos os casos são utilizados dispositivos "vídeo" que permite a visualização de imagens, assim como de um "computador" responsável pelo processamento do jogo. Portanto não é possível a dissociação entre videojogos e jogos de computador.

Um jogo electrónico, normalmente, é um conjunto de imagens de um mundo virtual de duas ou três dimensões, que tem como personagem um humano, animal ou veículo controlado pelo jogador. É por isso uma experiência interactiva que fornece ao jogador sequências de regras com desafios crescentes, que serão aprendidos, e eventualmente dominados pelo utilizador. Este estímulo de aprendizagem e de domínio são o elemento principal do que chamamos diversão, assim como uma anedota só tem piada quando é apreendida a sua lógica (Koster, 2013).

Na maioria dos jogos digitais, subsistemas do mundo real são modelados matematicamente de modo a serem manipulados por um computador. O modelo é uma aproximação e uma simulação da realidade (mesmo sendo uma realidade *imaginária*) pois é impossível importar para um mundo virtual cada detalhe do mundo real.

Todos os jogos electrónicos interactivos são simulações temporais, significando que o mundo virtual é dinâmico, ou seja, o estado do mundo do jogo é alterado ao longo do tempo enquanto a história e os eventos do jogo se desenrolam. Estes devem responder a interacções imprevisíveis do jogador humano em tempo real, fazendo deles simulações interactivas em tempo real (Gregory, 2009).

Os jogos electrónicos podem pertencer a diversas categorias, em que estas são depois divididas em infindáveis subespécies, existindo assim um número interminável de categorias. O seu género depende de vários aspectos a serem considerados, tais como o estilo ou o modo de jogabilidade, mas por ser algo tão diverso e indefinido os jogos podem pertencer a mais que uma categoria.

Apesar da maioria dos jogos terem, de alguma forma, uma componente educacional, os jogos educativos são claramente desenhados com esse propósito, e é nesse contexto que esta dissertação se insere.

2.1.1 Jogos Electrónicos Sérios

> *"O professor alerta que há dois pontos fundamentais no desenvolvimento de jogos electrónicos educativos: a interface e a didáctica. A comunicação homem-máquina tem de ser amigável, intuitiva e precisa. Para isso, o designer precisa encontrar so-*

> *luções que facilitem a acção do usuário. Ele afirma que os objectivos educacionais não podem ser explícitos, caso contrário, o usuário perde o interesse. "Um bom jogo não declara a sua proposta educacional de imediato. O ideal é que o aluno tenha vontade de jogar e, sem perceber, aprenda mais sobre a cultura europeia tentando pegar a ladra Carmen San Diego", sugere." (Tavares, 2013)*

As crianças e os jovens são introduzidos ao mundo tecnológico através dos videojogos, e a forma como eles interagem com os videojogos está a alterar a maneira como aprendem. O envolvimento e o compromisso são benefícios interessantes na utilização de jogos, mas nem todos têm um propósito educativo. O conteúdo de um jogo pode representar uma simplificação da realidade, e muitos jogos são baseados em temas violentos ou bastante sensíveis (Gros, 2007). Daí que a introdução dos jogos electrónicos em contexto escolar tem causado alguma controvérsia, pois são normalmente associados a comportamentos violentos. Contudo é conclusivo que os videojogos contribuem para a sociabilidade entre os jovens e aumentam a sua capacidade de liderança na tomada de decisões, para além de oferecerem um complexo ambiente de desenvolvimento de aptidões e atitudes (Johnson, 2006).

Ao longo dos tempos o desenvolvimento da tecnologia digital e do *e-learning*[2] tem contribuído para a sua difusão dos jogos electrónicos no processo de ensino e aprendizagem. Consequentemente, aumentaram os recursos disponíveis no que diz respeito a plataformas *e-learning* e ferramentas de ensino multimédia e são agora bastante aceites para a sua utilização nas escolas. O *e-learning* tem um grande potencial para a distribuição de meios de ensino de qualidade, facilitando a aprendizagem possibilitando novas perspectivas e abordagens ao conteúdo a ser transmitido (Kickmeier-Rust et al., 2007).

Os jogos correspondem a duas situações, são experiências activas e têm a capacidade de motivar as pessoas. Deste modo, estes estimulam e motivam a aprendizagem do jogador de uma forma apelativa e agradável, conseguindo um meio de aprendizagem que proporciona uma experiência positiva, usando um terminal de processamento de dados, normalmente computadores ou consolas. Os videojogos criam experiências em que os jogadores são inseridos em situações que os obriga a pensar e têm ferramentas e recursos com o objectivo de resolverem problemas complexos (Squire, 2005).

Assim, consideram-se os jogos úteis elementos para o ensino de estratégias específicas e para obtenção de conhecimento. Para além disso, também desenvolvem o conhecimento caracterís-

[2] Modelo de ensino não presencial suportado por um ambiente *online.*

tico da actual sociedade da informação, o que se prevê que traga benefícios a longo prazo, pois a interacção das crianças com os videojogos ajuda-os a terem uma aproximação lúdica com o computador, o que facilita a utilização e conhecimento da tecnologia (Gros, 2007).

Muitas das aplicações informáticas, especialmente os jogos de computador, têm especificidades que alteram a forma como é processada a informação, de verbal para visual. Nos jogos de acção, que têm sequências espaciais e dinâmicas assim como representações icónicas, existem acções a decorrer em diferentes locais. A capacidade que uma criança desenvolve ao jogar este tipo de jogos pode-lhes servir como preparação para as áreas da ciência e da tecnologia, onde a sua actividade depende da manipulação de imagens num monitor.

Outra capacidade incorporada neste tipo de jogos é a necessária divisão de atenção, pois o jogador tem que se manter, constantemente, a par de um grande número de acções que decorrem em simultâneo no ecrã. Um estudo feito por (Greenfield et al., 1994) explorou o efeito que os videojogos têm na divisão de atenção visual, estudando um grupo de alunos universitários. Os participantes que eram peritos em jogos de computador obtiveram melhores tempo de resposta do que os iniciados. Para além disso, ao jogarem jogos de acção os alunos desenvolveram estratégias para acompanhar diferentes eventos em vários locais. Resumindo, o estudo mostrou que os jogadores mais experientes desenvolveram uma melhor capacidade de atenção (Gros, 2007).

Um outro estudo foi efectuado por (McFarlane, Sparrowhawk, & Heald, 2002), testado com alunos do primeiro e segundo ano. O estudo foi baseado nas opiniões dos professores sobre as limitações e o potencial dos videojogos. O seu resultado reflecte que a maioria dos docentes reconhecem que os jogos contribuíram para o desenvolvimento de um largo número de competências que são bastante importantes: resolução de problemas, reconhecimento de sequências, capacidade de dedução e memorização. Além disso, também estratégias de grupo, como trabalho cooperativo e aprendizagem baseada em exercícios, podem ser facilmente inseridas no cenário de jogo (McFarlane et al., 2002).

As opiniões proferidas pelos docentes no estudo de McFarlane na Grã-Bretanha coincidem com as que foram obtidas dos professores chilenos, entrevistados por Nussbaum, demonstrado na Tabela 2.1 tabela.

Tabela 2.1 - Áreas em que o ensino utilizando jogos electrónicos pode contribuir (Nussbaum, Rosas, Rodríguez, Sun, & Valdivia, 1999)

Áreas	Aspectos em que os videojogos podem contribuir
Desenvolvimento pessoal e social	Proporciona interesse e motivação para aprender; Mantêm a atenção e concentração.
Linguagem e alfabetização	Encoraja a criança a explicar o que está a acontecer; Utilização da linguagem para organizar, sequenciar e clarificar ideias, sentimentos e eventos.
Desenvolvimento matemático	Aplicação de conteúdo matemático básico, focando na aritmética e geometria.
Desenvolvimento criativo	Utilizam a imaginação no *design,* na música e nas histórias.
Desenvolvimento físico	Aperfeiçoamento da capacidade motora na utilização de um rato ou *joystick* para navegação e selecção de objectos.

Os jogos educativos são também caracterizados por terem um ambiente de jogo visualmente atractivo, com música e animação, por despertar um interesse na exploração de jogo e cativar a atenção do jogador, por ser executado em tempo real e o jogador ter um *feedback* imediato das suas acções, assim como a existência de uma interacção simples e intuitiva entre jogador e sistema (Moratori, 2003). Este género de jogo é jogado com a finalidade do seu jogador explorar uma determinada área de conhecimento, além de desenvolver algumas capacidades, tais como destreza, reflexos, raciocínio lógico, entre outras (Chan & Ahern, 1999).

Os factores necessários para a escolha de um jogo derivam do jogador alvo e da realidade pretendida. É importante entender a capacidade de aprendizagem do jogador, havendo diferenças no modo como processam informação e das suas emoções aquando da exposição ao mesmo problema. Estas diferenças podem derivar de diversos factores, tais como a realidade social ou a sua idade, entre outros. É necessário também ter em atenção quais as habilidades que se desejam desenvolver no jogador, sejam estas cognitivas ou físicas (Rapeepisarn, Wong, Fung, & Khine, 2008).

2.1.1.1 SimCity

É um jogo de simulação que permite ao jogador construir e desenvolver uma metrópole virtual. No SimCity o jogador toma todas as decisões, desde o tipo de plantas a ser cultivado e onde, às zonas industriais, comerciais e residenciais. Apesar de ter um aspecto bastante intuitivo e simplista este é um jogo bastante complexo, onde cada acção custa dinheiro. É atribuindo um *cachet* no início do jogo, e o dinheiro aumenta ou diminuí ao longo do jogo através das acções

efectuadas pelo jogador, tais como a cobrança de impostos, a exploração de minas petrolíferas, entre muitos outros.

A interacção entre o jogador e o computador é constante e o intuito do jogo não se baseia apenas na tomada de decisões, como também em proporcionar a possibilidade de explorar os processos sociais, práticos e económicos na gestão de uma cidade."SimCity é uma forma inovadora de nos tornarmos empreendedores e conseguir responder aos desafios impostos pela sociedade" (Friedman, 1999).

Devido à utilização em larga escala do SimCity nas escolas e em casas particulares é fácil encontrar elementos que comprovam que este jogo é uma melhor e mais influente introdução ao planeamento urbano do que qualquer livro, como fez Paul Starr, fundador do American Prospect[3].

Apesar de serem ainda significativas, as diferenças entre este jogo e uma cidade real tendem a diminuir, pois o jogo evoluiu, após várias versões, de forma a considerar aspectos mais sofisticados da vida real. Além deste facto, a ligação do planeamento urbano e do SimCity é ampliada pelas ferramentas utilizadas pelos urbanistas, pois estas assemelham-se bastante com uma perspectiva *top-down* deste jogo.

Figura 2.1 - Ecrã de jogo do SimCity

[3] É uma revista política bimensal, situada em Washington D.C., EUA.

2.1.1.2 Sid Meier's Civilization:

Civilization é um jogo de estratégia e simulação que foi promovido como ferramenta educacional, utilizado na construção de simulações educacionais e que tem como objectivo a evolução de uma civilização histórica, desde o ano 4000 A.C. até à actualidade, sendo um dos *franchises* de videojogos mais bem-sucedidos e tem sido utilizado como suporte educacional para as escolas e como modelo para construção de jogos que retratem a evolução histórica (Lane, 2007).

Este jogo foi considerado como uma ferramenta educacional que ajuda os jogadores a terem conhecimentos sobre aspectos históricos, políticos, económicos e militares. Ensina a valorizar a cultura, os aspectos geológicos e históricos (Friedman, 1999).

Começando apenas com um guerreiro, constroem-se cidades, estradas, templos, exércitos, entre outros, ao longo de milhares de anos. Após diversas versões, hoje é possível o jogador triunfar através de meios culturais, diplomáticos e científicos, invés de apenas força militar. Isto permite ao jogador escolher entre vários objectivos e tácticas possíveis de forma a derrotar os restantes jogadores, que têm os mesmos objectivos controlando outras civilizações. Enquanto vencer através de meios militares ou científicos pode ser simples, uma vitória diplomática é considerada pela comunidade de jogadores de Civilization como um feito quase heróico (Owens, 2011).

Como é um jogo que tem um rumo dependente das acções do jogador, pois é ele quem decide os aspectos a desenvolver, ou que peças serão criadas ao longo da história do jogo, então numa análise mais detalhada poderiam ser apontadas diversas falhas e problemas no modelo da história da ciência do jogo. Contudo, o reconhecimento dessas falhas e problemas apontados ao modelo faz com que os jogadores tenham uma concepção mais profunda do seu próprio conhecimento acerca da ciência e da tecnologia (Apperley, 2010).

Figura 2.2 - Ecrã de jogo do Sid Meier's Civilization V

2.1.1.3 Bot Colony (2013):

"We have over 7,000 people from 176 countries in our database who have all signed up for the beta just to practice their English. We're hopeful that this won't be just a game, but a fun way for them to improve their lives." (Joseph, 2014)

Figura 2.3 - Imagem de publicidade do Bot Colony

Bot Colony é baseado num livro de Eugene Joseph, com o mesmo nome. É um jogo de aventura por episódios *online* e individual desenvolvido pela North Side.

O jogador controla a sua personagem e tem de estabelecer contacto com *robots*, através de um *tablet*, de forma a obter informações para completar as missões. Este contacto é feito através de um diálogo inteligente em inglês, entre o jogador e os personagens não controlados pelos jogadores. A maior inovação deste jogo é a capacidade de comandar um personagem, ou fazer uma questão relevante, vocalmente e em inglês, e obter uma resposta ou uma acção condizente.

O jogador comunica com o jogo falando através de um microfone e o jogo utiliza um *software* de discurso-para-texto para interpretar e reagir ao que foi falado. É também possível a comunicação ser feita através da digitação em inglês, caso o computador não reconheça as palavras que foram ditas através do microfone.

North Side pensa em Bot Colony como uma ferramenta no ensino de Inglês como segunda língua e não para aprendizagem de inglês como língua materna. Este jogo oferece a oportunidade de ter uma experiência com um *robot* semelhante ao diálogo com um professor humano inglês, pois este corrige erros e fornece *feedback* durante o diálogo. É bastante útil para jogadores com problemas de pronunciação e para outros que poderão também praticar a escrita.

Bot Colony promove a aprendizagem do Inglês de uma forma prática e em situações reais onde seja preciso falar inglês para lidar com o dia-a-dia (Joseph, 2014).

Player: I need a taxi?
Tina: I can confirm that taxis are not responding.
Tina: However, the Hunter-Bot can carry passengers.
Therefore, the Hunter-Bot could carry you to your destination.
Only authorized robots can open the door to the Hunter Bot's room.
Player: Who is authorized to open the door?
Tina: Only two robots are authorized to enter the Hunter-Bot room.
Player: what robots have clearance?
Tina: Maintenance-Bot "Kenny" MT-133 and Vacuum-Bot "Charlie" VC-009 can enter the hunter bot room.
Player: where is Kenny?
Tina: The Maintenance-Bot "Kenny" is in the Robot Maintenance Facility for a system upgrade.
Player: where is the robot maintenance facility?
Tina: I don't know where it is.
Player: Where is Charlie?
Tina: The vacuum bot "Charlie" is currently working on the lower level.
Player: thanks
Tina: You're very welcome.

Figura 2.4 - Exemplo de conversa interpretada do Bot Colony

2.1.1.4 Tetris

Tetris é um jogo inventado pelo matemático Alexey Pazhitnov nos anos 80 e é um dos jogos computorizados mais distribuídos de sempre. É bastante popular e semelhante a um *puzzle*, onde a cada jogador é dada uma sequência aleatória de peças, com o intuito de este as encaixar na forma de um tabuleiro rectangular. Ao completar uma fila esta desaparece e todas as peças acima de si caem. O objectivo é completar o máximo de filas possíveis antes que o tempo se esgote (Demaine, Hohenberger, & Liben-Nowell, 2003).

O jogador sabe antemão qual a próxima peça a sair perde quando o tempo acaba ou quando as peças se empilham até ao topo. Hoje em dia é possível jogar Tetris não só nos computadores, mas também em consolas, telemóveis, calculadoras gráficas, *tablets*, entre outros.

Através do Tetris é possível melhorar as funções cognitivas, como o pensamento crítico, raciocínio, linguagem e processamento e a rapidez de pensamento (Demaine et al., 2003).

Figura 2.5 - Ecrã de jogo do Tetris

2.1.1.5 Math Blaster

O sistema de aprendizagem **Blaster** é uma série de videojogos com bastante sucesso de vendas, originalmente criados pela Davidson & Associates em 1983, com o jogo original chamado Math Blaster, produzido para vários sistemas de computadores, consolas e dispositivos portáteis. Originalmente a série apenas ensinava matemática mas mais tarde expandiu-se para a ciência e para a leitura.

No primeiro jogo o personagem principal era Blasternaut e era acompanhado por um amigo numa viagem espacial. Enquanto faziam reparações na sua nave o seu amigo é raptado e o jogador é transportado para uma planeta que serve de base a *aliens*. Nesse planeta, Blasternaut tem que ultrapassar uma série de obstáculos e desafios, à base de somas, adições, multiplicações e divisões, de forma a derrotar os *alien* e salvar o seu amigo (Wiki).

Mais tarde, com a tecnologia disponível muito mais avançada que na altura do primeiro jogo, foi desenvolvida nova versão, melhorada graficamente e com novos e imensos desafios diferentes adicionados, em que o jogador é um comandante galáctico preso num planeta de macacos. Para ajudar o comandante a escapar o jogador tem que obter medalhas, que são conseguidas através da resposta correcta a vários desafios matemáticos, como adições, subtracções, multiplicações e divisões, com números inteiros, decimais, fraccionais e probabilidades conseguindo assim conquistar a atenção dos jogadores através da utilização de animações, sons, uma história interessante e de personagens animados para o ensino da matemática (Rodrigo, 2010).

Actualmente, o jogo foi expandido para aplicações móveis, com cinco aplicações matemáticas e com um sistema de jogo *online*, que inclui um sistema de “amigos”, em que os alunos podem interagir com outros, e executarem desafios juntos (Gallagher, 2012).

Figura 2.6 - Ecrã de jogo do Math Blaster

2.2 Motores de Jogo

"Um motor de jogo é uma framework que consiste num conjunto de diferentes ferramentas, utilidades e interfaces que ocultam os detalhes mais específicos de várias tarefas que compõem um jogo de vídeo" (Sherrod, 2006)

A distinção entre um jogo e o seu motor é, por vezes, bastante confusa e vaga, assim como a própria definição de motor de jogo. Alguns motores de jogo têm uma clara distinção entre o seu jogo, enquanto outros praticamente não se consegue decifrar a diferença entre um e outro. Num jogo, o código de renderização[4] poderá estar especificamente definido de forma a desenhar um objecto na perfeição, enquanto noutro jogo, o motor de renderização poderá fornecer materiais que podem ser aplicados em vários objectos diferentes, sendo possível desenhar o mesmo objecto totalmente a partir de dados.

Possivelmente uma arquitectura orientada por dados é o que distingue um motor de jogo de um *software* que seja um jogo e não um motor. Quando um jogo tem as suas regras ou lógica programadas por grandes extensões de código, ou que aplique casos específicos de código para renderizar objectos exclusivos, então torna-se bastante complicado, ou até impossível reutilizar esse *software* para criar um outro jogo que não o que se destina. O termo **motor de jogo** deverá ser reservado para *software* que seja extensível e que possa ser utilizado como base de diferentes jogos sem grandes alterações (Anderson, Engel, Comninos, & McLoughlin, 2008). Um motor de jogo é assim uma *framework* com uma vasta biblioteca, que suporta a introdução de *scripts* e objectos de acordo com as ideias do jogo, assim como toda a configuração de controlo de informação, independentemente de ser um jogo 2D ou 3D ou do seu género.

Os jogos são naturalmente aplicações multimédia em que o *input* de dados chega ao motor de jogo de variadas formas, desde objectos 3D, imagens em mapas de *bits,* animações e ficheiros de áudio. Todos estes recursos deverão ser criados e manipulados pelos respectivos artistas (Anderson et al., 2008).

Hoje em dia, os jogos são construídos de forma modular, então o motor de jogo refere-se ao conjunto de módulos de simulação de código. Estes *engines* são constituídos por módulos responsáveis pelo *input*, *output* e pela dinâmica genérica do mundo de jogo, como é demonstrado, através de um modelo simplista, na Figura 2.7.

[4] Transformação pela qual se obtém o produto final de um processamento digital

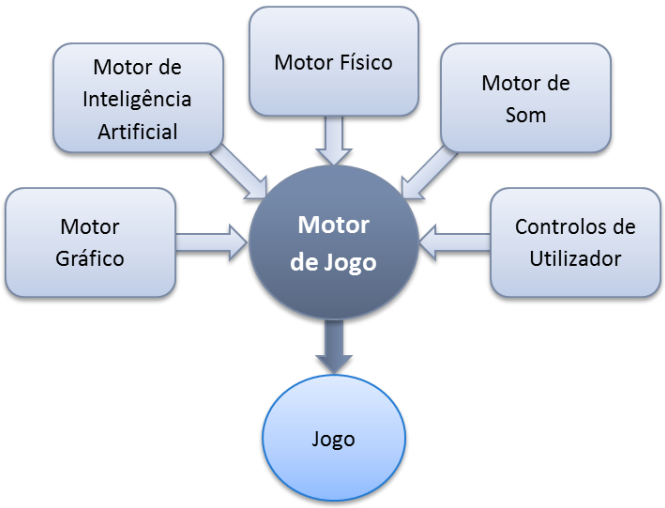

Figura 2.7 - Estrutura de motores de jogo modulares (Nilson & Söderberg, 2007)

- Os gráficos são extremamente importantes em qualquer jogo, pois são através deles que é causada a primeira impressão. O motor gráfico é o responsável por efectuar todo o processamento matemático de forma a permitir que o jogador visualize o jogo. Devido ao avanço na área tecnológica, mais precisamente na maior capacidade do *hardware* disponível nos computadores, houve um aumento na complexidade destes cálculos gráficos utilizados nos jogos.

- Um motor de Inteligência Artificial está a tornar-se um elemento cada vez mais importante na indústria dos videojogos. Pode-se dizer que é o cérebro do jogo. A forma como o jogo de computador reage e responde aos *inputs* do utilizador é produzida pela inteligência artificial. Durante o jogo também os inimigos e os personagens não controlados pelo jogador utilizam inteligência artificial para tomarem decisões.

- Os jogos são muitas vezes inspirações, ou até mesmo representações do mundo real e um dos aspectos fundamentais do universo é a física. Apesar da física contribuir para um videojogo oferecendo uma experiência mais imersiva, só recentemente é que os jogos começaram a utilizar físicas mais realísticas. Isto foi principalmente motivado pela enorme capacidade que é imprescindível para efectuar cálculos físicos que tornariam o jogo mais lento e pesado, mas com o aumento do desempenho do *hardware* hoje em dia é possível obter físicas bastante aproximadas da realidade. Então o motor físico é o executor de todos estes trabalhosos cálculos, permitindo que os objectos do jogo se comportem de forma mais realista.

- O som é uma parte tão importante num jogo como é num filme, excepto que num jogo este pode ser interactivo, o que o torna ainda mais complexo. A função do motor de

som é reproduzir os dados de áudio que são solicitados, sem causar nenhum atraso em nenhuma das outras funções do jogo. Isto quer dizer que os ficheiros áudio têm que estar em memória e prontos a serem reproduzidos porque possivelmente o carregamento directo dos sons do disco rígido é mais demorado.

- Os controlos de utilizador são a maneira que o jogador tem de interagir com o jogo. Esta pode ser feita com um variado número de aparelhos, normalmente com o rato, o teclado ou um *joystick*. Hoje em dia existe também a possibilidade desta interacção ser feita através de outros dispositivos, como sensores de movimento ou de pressão.

- O motor de jogo, visto da perspectiva em que é o responsável pela união e interacção de todos os outros motores, é então onde todas as categorias como gráficos, recursos como imagens e sons do departamento sonoro, bases de dados, etc. se integram. Pode-se dizer que é a parte mais importante de qualquer jogo. No entanto, se nenhum dos restantes motores estiver disponível este bloco central resume-se a zero. Então, todos os componentes estão fortemente interligados e dependentes uns dos outros (Busby, Parrish, & Wilson, 2009).

A escolha do motor de jogo é uma questão que se põe para o desenvolvimento de um jogo, pois este terá que ser apropriado para o objectivo desejado. Assim, foram considerados algumas ferramentas utilizadas para a criação de jogos, entre elas, o Microsoft XNA, Unreal Engine, Unity®, Cry Engine, Panda3D, Blender, Exult, Godot, Kobold2D, Magnum e Ogre3D. Foi feito um estudo às suas características e capacidades, considerando vários aspectos e de entre as inúmeras que existem disponíveis no mercado foram seleccionadas quatro das mais utilizadas:

- **Unreal Engine –** O UnrealEngine é um motor de jogo, inicialmente utilizado para jogos de tiro em primeira pessoa, que tem incorporado vários sistemas, ferramentas e atributos.

- **GameMaker: Studio –** É um sistema de criação de videojogos multiplataforma utilizando um sistema de *drag-and-drop*, ou de uma linguagem GML para jogos mais complexos.

- **Cry Engine –** CryEngine foi inicialmente desenvolvido para ser aplicado como uma demonstração de tecnologia e mais tarde, dado o seu potencial, este foi expandido e daí resultou um jogo .

- **Unity® –** O Unity® é um sistema utilizado na criação de jogos, multiplataforma, que tem incluído um motor de jogo e um IDE.

2.2.1 Unreal Engine

O Unreal Engine foi concebido para criar jogos FPS e é basicamente um sistema que organiza os recursos do jogo, como os personagens, músicas, efeitos de som, gráficos, etc, num ambiente visual interactivo. Este contém diversos componentes que podem funcionar independentemente uns dos outros, mas que são mantidos em harmonia por um núcleo central. Graphics Engine, Physics Engine, Input Manager e o UnrealScript Interpreter são alguns dos componentes pertencentes ao Unreal Engine (Busby et al., 2009).

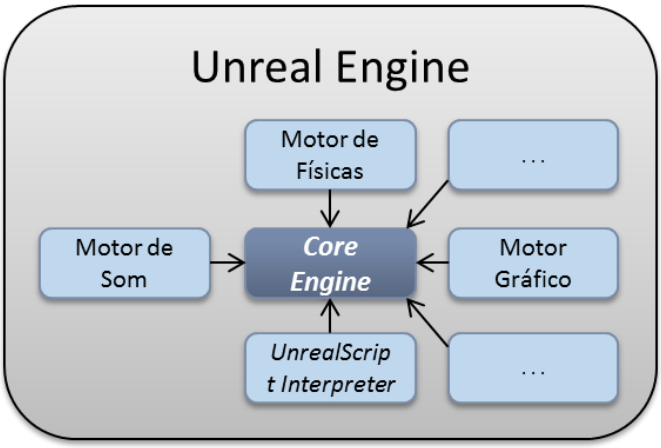

Figura 2.8 - Unreal Engine

Este motor de jogo suporta OpenGL e DirectX10, e permite a criação de jogos para PC, Xbox360, PS3 e iOS. Oferece também ferramentas para utilização de luzes dinâmicas, sistema de rede, motores de física, *shaders*, sombras, reprodução de sons, animações usando AnimTrees, motor de físicas, entre outras (Muñoz Rueda, 2012).

2.2.1.1 Motor Gráfico

Este motor de gráficos é o responsável pela transmissão de imagens ao jogador no decorrer do jogo. Ele executa processos que decorrem sem que seja visível ao jogador, mas que calcula quais as imagens a serem exibidas e quais devem ser ocultadas. È neste componente que também são controlados todos os aspectos visuais dos objectos do jogo, transformando linhas de código e variáveis em imagens.

O Unreal Engine tem a particularidade de quando os objectos são escondidos por trás de outros ele não os interpreta, libertando assim memória para processos mais úteis e urgentes.

Este motor também suporta Level Streaming, que permite que, durante o jogo, sejam carregadas e descarregadas as texturas de diferentes subníveis da memória. Isto quer dizer que

quando o jogador entra num cenário ou nível específico todas as texturas da área anterior desaparecem da memória e o Graphics Engine deixa de as interpretar, permitindo assim ao jogador andar em cenários vastos sem haver quebras para processamento de imagens (Busby et al., 2009).

2.2.1.2 Motor de Físicas

O Physics Engine é o responsável por transportar o mundo de colisões e leis da física que vemos todos os dias para o computador, mais precisamente para o jogo. Ele é o responsável pelas explosões, colisões, fragmentação de objectos, entre outros que existem nos jogos. Apesar de ser possível criar todo este mundo de interacções entre objectos através de programação, o Unreal Engine facilita o trabalho utilizando um motor de físicas chamado PhysX, criado pela NVIDIA, que já tem incluído um sem-fim de simulações físicas ao dispor do utilizador (Busby et al., 2009).

2.2.1.3 Gestor de *Inputs*

Sem os *inputs* do jogador não existe um jogo, pois sem a sua interacção ele passa a ser um espectador de uma animação. Estes *inputs* chegam ao computador através de movimento do rato, controlos analógicos, botões ou até de dispositivos de captura de som ou movimento.

O Input Manager tem o objectivo de reconhecer os *inputs*, independentemente da sua fonte, e ao configurar um conjunto de propriedades converte-os num comando que designa qual será a reacção do jogo (Busby et al., 2009).

2.2.1.4 UnrealScript Interpretor

UnrealScript é um dos aspectos mais inovadores do Unreal Engine. É uma linguagem de programação bastante semelhante a Java ou C++, mas com a intenção de ser relativamente mais simples que estas. Os *scripts* permitem aos utilizadores ajustarem os processos que o motor de jogo executa, sem ser realmente necessário programar o código fonte do jogo.

O UnrealScript Interpretor é o componente responsável por transformar o código escrito em UnrealScript para código que o motor consigo processar (Busby et al., 2009).

2.2.1.5 Interacção entre componentes

O Unreal Engine utiliza um *game loop* dependente de acontecimentos. Isto quer dizer que o motor tem uma lista de eventos que necessita de executar, que lhe são transmitidos a partir dos vários componentes. Ao invés de actualizar todos os aspectos do jogo durante cada cena, este motor de jogo atribui uma prioridade específica a esses eventos.

Quando o jogo é inicializado são carregados todos os *engines.* Após isso todos os componentes começam a comunicação entre eles, sendo o *core engine* o responsável pela inicialização de todos os outros módulos, enviando informação sobre o estado inicial de todos eles. Uma vez que todos os componentes estejam carregados, é apresentado o UI do jogo e o *core engine* fica a aguardar o *input* do jogador. Para que seja possível o início da parte jogável o *core engine* carrega um nível que contém todos os recursos necessários para o *gameplay*, enviando-os para os respectivos *engines*. A lista de eventos é então preenchida e é responsabilidade do *engine core* geri-los de forma a executar o jogo (Busby et al., 2009).

Figura 2.9 - *Game Loop* do Unreal Engine

2.2.2 GameMaker: Studio

GameMaker: Studio é uma ferramenta poderosa e de fácil utilização, que utiliza o conceito de *design* orientado a objectos, para a criação de jogos, com um ambiente de desenvolvimento integrado destinado a jogos bidimensionais. Foi inicialmente desenvolvido com o intuito de ser usado em ambiente escolar, para os alunos compreenderem conceitos de programação básica, entenderem a arquitectura dos jogos e criarem animações 2D. Esta ferramenta tem uma linguagem de programação bastante robusta, tem a possibilidade de importar recursos gráficos e sonoros, e a integração de uma biblioteca de físicas 2D, apesar de não existir um motor de físicas (Elliott, 2013).

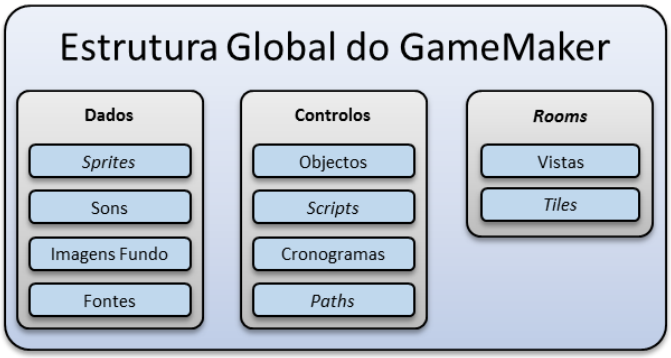

Figura 2.10 - Estrutura Global do GameMaker

Este motor de jogo é focado para jogos de duas dimensões, com um motor gráfico baseado em *sprites*[5]. Contudo tem também algumas funcionalidades 3D, embora limitadas, e tem um interface intuitivo de *drag-and-drop*, que evita o recurso à programação, apesar de existir uma linguagem própria, Game Maker Language, ou GML, quando necessário (Overmars, 2009).

No GameMaker é possível definir objectos, quer tenham representações visuais ou não, e a criação de várias instâncias do mesmo objecto, assim como a definição das suas hierarquias. Estas instâncias têm propriedades que podem ser definidas, manipuladas e verificadas usando acções ou código. O comportamento desses objectos é definido pelo utilizador, não sendo possível que um objecto existente num jogo seja partilhado noutro. Podem ser partilhados recursos, como imagens e sons, mas não objectos completos.

O GameMaker tem um comportamento baseado em eventos. Os eventos ocorrem num determinado objecto, e o *designer* especifica qual a acção que o jogo tem a executar tendo em conta esse evento, ou conjunto de eventos. Por norma esses eventos são a criação ou destruição de objectos, *inputs* do utilizador ou colisões entre objectos. Para se alcançar um certo comportamento, o *designer* pode simplesmente arrastar acções para dentro dos eventos, utilizando o simples interface do motor de jogo. Estes procedimentos utilizam um motor lógico, que pode ser executado a partir de acções ou através de código (Bailey & Games, 2011).

Este motor de jogo executa numa escala de *steps* discretos, que podem ser vários por segundo, normalmente entre 10 e 200. Todos os objectos executam de forma concorrente, e por cada *step* ocorre a sequência de eventos seguinte:

[5] Colecção de imagens de jogo que armazenam uma animação de alguma objecto do jogo

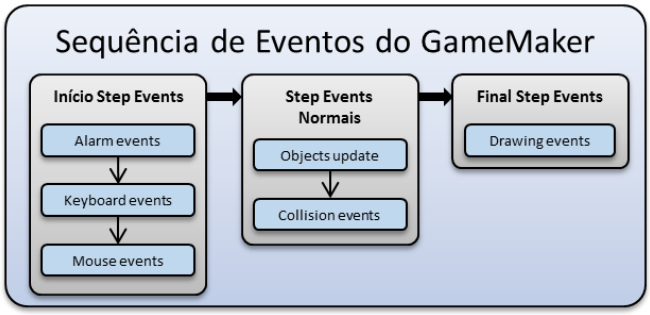

Figura 2.11 - Sequência de um *step* do GameMaker

Estes ***step events*** são oportunidades para ser executado o código criado pelo programador, que ao contrário de outros motores de jogo permite ao criador ter controlo da temporização e a capacidade de assegurar que acções específicas sejam executadas antes de outras com um único *step,* e também controlar a execução de código em relação a uma sequência fixa de eventos. A temporização dos eventos é definida pelo programador, que pode ser múltiplo do *clock* do motor de jogo, e podem ser accionados em tempos específicos em qualquer uma das três fases de *step events*.

Os aspectos únicos que existem no GameMaker permitem a que utilizadores inexperientes criem jogos, pois o sistema de *drag-and-drop* é bastante mais simples que a programação de scripts complexos. Contudo, o GameMaker basicamente permite alterar o visual, alterar as regras utilizando um sistema intuitivo e editar as propriedades de objectos do jogo, combinando esses objectos com regras, num *template* de um jogo, limitando assim a personalização e especificação de conteúdo.

Existe uma versão gratuita do GameMaker, versão Lite, e uma versão paga, com funcionalidades completas, ambas disponíveis para *download* no site oficial da YoYoGames.

2.2.3 CryEngine

O CryEngine é um motor de jogo construído num nível acima de uma *sandbox*[6] que suporta um número de características que permitem criar áreas de terreno realístico, enormes e imersivas. Este motor tem ferramentas de desenvolvimento integradas, tais como um editor real-time, um sistema de física, sistema de música, sistema de rede, *shaders*,que é a produção de vários níveis de cor numa imagem, iluminação dinâmica e *bump mapping*, que consiste num mapa de altura que faz variar a intensidade da luz.

[6] Jogo com jogabilidade não-linear em que os seus desafios podem ser completados num número de sequências diferentes.

As vantagens para a sua utilização são os cenários com gráficos de muito alta qualidade que se podem produzir com este motor. Essa enorme capacidade gráfica, por sua vez, impõe também a existência de *hardware* de altíssimo desempenho (Trenholme & Smith, 2008).

Este motor de jogo tem uma interface de programação intuitiva baseada em nós, que permite controlar o decorrer dos eventos de forma visual, evitando assim ter que recorrer à programação. Tem também um sistema de partículas em tempo real que permite a criação de complexas explosões e colisões em poucos passos, assim como um sistema de efeitos especiais. Como o seu maior foco é o aspecto gráfico, tem poderosas ferramentas incluídas que permitem a rápida criação de terrenos e água, usando um forte sistema de *shaders*, aliado a uma ferramenta de iluminação que proporciona um excelente ambiente visual. Inclui também um sistema de edição de áudio bastante eficiente, que permite a edição de sons de forma interactiva. As aplicações que utilizam este motor operam em computadores com sistema operativo Windows, PS3 e Xbox360.

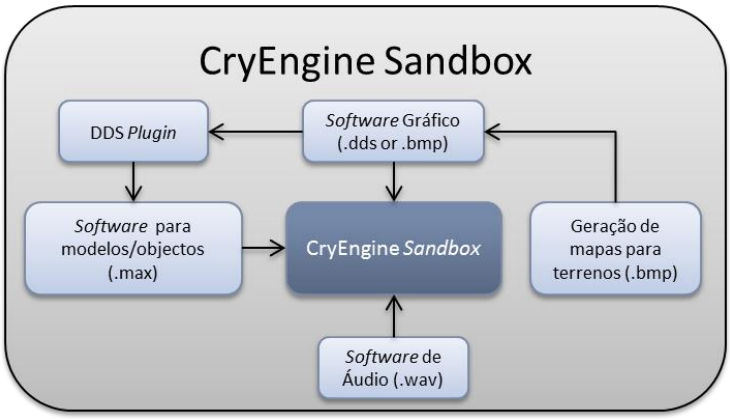

Figura 2.12 - CryEngine *Sandbox workflow*

2.2.4 Unity®

O Unity 3D, também conhecido apenas como Unity®, é um ecossistema completo de ferramentas e serviços criados que facilitam o trabalho aos programadores de jogos e de conteúdo interactivo multiplataforma. É um motor de jogo, com um editor totalmente integrado (IDE) que permite a criação de jogos de modo simples e rápido, tanto 3D como 2D, e possibilita que estes sejam executados em diferentes ambientes. Este IDE do Unity® permite que as "peças visíveis" do jogo possam ser reunidas, utilizando um sistema de *drag-and-drop,* e esta edição é feita em conjunto com uma pré-visualização do resultado final.

O editor de jogo Unity® tem também um motor de *scripts* próprio, baseado em Mono[7], que permite especificar acções, efeitos ou comportamentos nos objectos, chamado MonoDeveloper. Este motor permite a programação em diferentes linguagens como C#, JavaScript e Boo, que é um dialecto baseado em Python. O código máquina gerado é baseado na plataforma .NET e permite a importação de DLL's baseadas também na plataforma .NET e criadas em C++ .

O Unity® suporta diversos formatos multimédia, tais como mp3, jpeg, gif, mov, avi, wav, entre outros. Também suporta variados formatos de modelos 3D, e estes podem ser importados nos formatos nativos de programas como Maya, 3DS Max, Blender, Cheetah 3d, Photoshop, Fireworks, entre outros. Como sistema de renderização o Unity® utiliza várias plataformas, como o Direct3D e OpenGL, dependendo da plataforma para a qual é exportado o jogo (Creighton, 2010).

O sistema de colisões que o Unity® utiliza como motor de físicas, que efectua todos os cálculos físicos e de interacções entre objectos do jogo, é o motor PhysX, criado pela NVIDIA, que permite a criação de simulações físicas sem a necessidade de criar *scripts* próprios para esse fim.

Dada a complexidade desta ferramenta, é apresentada uma simplificação desta framework representando apenas os aspectos gerais, na Figura 2.13.

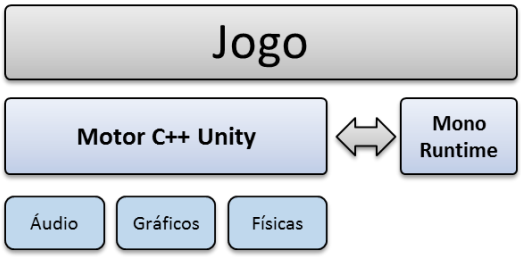

Figura 2.13 - *Framework* Unity 3D

Outro dos aspectos mais poderosos do Unity® é a sua capacidade de integração multiplataforma. Isto quer dizer que facilmente se consegue exportar um jogo para plataformas como PS3, Xbox360, Wii U, Windows, Mac, e também para *players web* tanto em Windows como em Mac.

Existem diferentes modalidades de licenças para o Unity®. Há uma versão do Unity® que, como ferramenta de desenvolvimento, é maioritariamente gratuita, sem comissões nem taxas e existe uma versão Pro, que custa $1500, que oferece alguns aspectos, para criação mais deta-

[7] Implementação de distribuição livre da *framework* .NET

lhada de jogos, que não existem na versão gratuita, como iluminação global, *render-to-texture*, entre outros (Viana, 2009).

É de relevante importância referir que o Unity® é o motor com mais expressão de mercado neste momento, com 45% de *share* em relação ao mercado de motores de jogo, enquanto o seu concorrente mais próximo apresenta 17%. Conta com 3.3 milhões de programadores de jogos registados, contando entre eles com Cartoon Network, Coca-Cola, Disney, Electronic Arts, etc (Unity Technologies).

2.3 Síntese

Este capítulo forneceu um enquadramento dos diferentes tópicos da dissertação, fazendo agora um resumo às principais características de cada elemento, relacionando-as com as características do problema. Assim, é possível entender que se vai desenvolver uma solução de forma a auxiliar a criação de conteúdos educativos. Esta solução será um jogo com fins educativos de forma a cativar os alunos com outra abordagem ao ensino que, como foi demonstrado, traria vantagens para a área da educação e, particularmente, para área de educação e desenvolvimento crianças com “diferenças”.

Foram também identificados vários motores de jogo, que podem ser utilizados como base do jogo. O recurso a estas *frameworks* permite a utilização de modelos detalhados existentes nas suas bibliotecas, evitando assim a criação de complexos componentes. As diferenças mais significativas entre motores de jogo podem ser observadas na Tabela 2.2.

Tabela 2.2 - Comparação de Características entre Motores de Jogo

	UnrealEngine	GameMaker	CryEngine	Unity
Multiplataforma	✓	✓	✓	✓
Sistema Operativo	Windows, Linux, Mac OS	Windows	Windows	Windows, Mac OS
Scripts	UnrealScript	GML	Lua	C#, JavaScript, Boo
Integração com web	×	✓	×	✓
Importação de conteúdo CAD	3ds Max; Maya	Através de outros softwares	Através de outros softwares	Blender, 3ds Max, C4D, Maya, Cheetah 3D, Zbrush, Photoshop, FireWorks
Físicas	Detecção de Colisões; Rigid Body; Vehicle Physics	Interaccção entre objectos; Apenas 2D	Detecção de Colisões; Rigid Body; Vehicle Physics	Detecção de Colisões; Rigid Body; Vehicle Physics
Shaders	✓	✓	✓	✓
Suporte Gráfico	3D	2D e 3D	3D	2D e 3D
Animações	✓	✓	✓	✓
Extensões / Plugins	✓	×	✓	✓
Comunidade	●●●●●	●●●○○	●●○○○	●●●●●
Licença	$19 / mês	Gratuita (Versão LITE)	$10 / mês	Gratuita (com limitações)
Classificação	●●●●○	●●●○○	●●●○○	●●●●●

3 Ferramenta Proposta

Neste capítulo será apresentado um jogo, que servirá como modelo de base à implementação da solução. Este modelo de jogo foi o seleccionado porque é focado para a área da educação, que é parte integrante do objectivo desta dissertação e será descrito para que se perceba o seu modo de funcionamento, assim como os requisitos existentes.

Esta apresentação é importante para que sejam depois identificados os aspectos necessários e as especificações técnicas a incluir na solução, para que se consiga a criação deste jogo e posteriormente é apresentada uma proposta de solução para a resolver os problemas identificados.

Com a implementação de um *Wizard* pretende-se criar uma simplificação de todo o moroso processo e etapas que fazem parte da criação de um jogo, a partir das características especificadas. A partir do modelo identificado podem-se criar diversos jogos, mantendo o mesmo mecanismo, mas é possível que siga outra história e que tenha um objectivo global diferente. Por exemplo, poderia ser utilizado este modelo de jogo, seguindo uma outra história, para o ensino de operações matemáticas.

3.1 Modelo de Jogo

3.1.1 História do Jogo

O jogo consiste num universo imaginário, que foi totalmente desenvolvido pelo centro **Diferenças**, chamado "**No Reino dos Fonemas**", e na sua família real que vivia no seu castelo, composta pelo Rei Fonos, a Rainha Xílaba e as cinco Princesas suas filhas que, curiosamente, o nome de cada uma delas começa com uma das vogais. A certa altura, de uma trovoada inesperada surge o Feiticeiro Errum, que entra pela janela e encontra toda a família real. Com um poderoso feitiço, feito com a sua varinha mágica, o feiticeiro fez desaparecer as Princesas. Para além do desaparecimento das filhas do rei, o feiticeiro Errum também levou consigo os baús repletos de tesouros do reino.

Com isto e sem poder abandonar o seu castelo, o rei lançou um pedido de ajuda em busca de jovens que tivessem coragem de aprender a derrotar os feitiços do feiticeiro para salvar as suas princesas.

Respondendo ao pedido do rei, apareceram dois jovens irmãos, a Alfa e o Berto, entusiasmados para assumir o desafio de vencer o mago e resgatar o tesouro do reino e as suas princesas. O jogador tem então de escolher qual o personagem principal com que vai jogar.

O Rei entrega então um mapa do reino ao herói, Figura 3.1, e informa-o sobre o percurso que deverá fazer para chegar à Torre Mágica, que se encontra na ilha Alfabeto, onde habita o feiticeiro Errum. Para lá chegar, o jogador terá que ultrapassar todos os obstáculos que enfrente na ilha Vogalis para ser levado de barco até à ilha das Consoantes. Para chegar até à ilha Alfabeto terá também que resolver todos os desafios propostos na ilha Consonantis.

Figura 3.1 - Reino dos Fonemas - Mapa do Reino

Na ilha Vogalis o jogador tem cinco desafios a realizar. No primeiro desafio, que é um jogo dinâmico, ele terá que percorrer três etapas em busca de balões com a vogal correspondente ao desafio, ou com um objecto em que a vogal surja no nome, para conquistar a moeda com essa vogal, como exemplificado na Figura 3.2. Com essa moeda o jogador é levado a outra série de quatro etapas. Na primeira o jogador tem que reconhecer o som da vogal e apanhar tudo o que tenha esse som. Pode ter de identificar quais as palavras correspondentes de entre várias diferentes, ou a sílaba que contêm o som da vogal numa palavra ou de entre vários objectos escolher o que contêm essa vogal. Depois de ultrapassado mais este obstáculo o jogador salva a princesa cujo nome começa com essa mesma letra. Os restantes quatro desafios são em tudo semelhantes com o primeiro, diferenciando em termos de cenário, das letras que aparecem nos balões, e claro, na princesa. Nestes níveis o jogador é acompanhado por um animal, cujo nome também começa pela respectiva vogal, que o aconselha com acções que o jogador deve tomar.

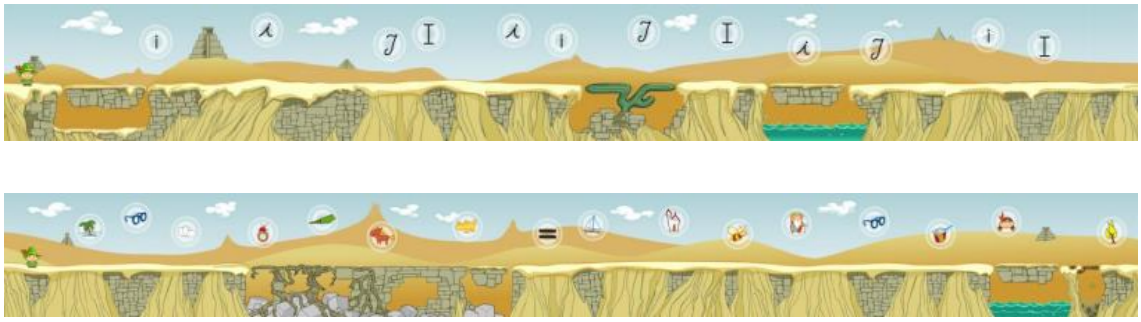

Figura 3.2 - Reino dos Fonemas - Exemplos de Níveis

Quando ultrapassados os obstáculos da ilha Vogalis o jogador recolhe todos os objectos e respectivas moedas e tem que inserir as moedas na sequência correcta das vogais para abrir o portão que o leva ao barco para a ilha Consonantis.

Na ilha Consonantis encontra-se o Duende Chefe, a quem foi lançado um feitiço que o fará transformar-se num gigante e enquanto é acompanhado até ao barco o vigia diz-lhe que é necessário que o jogador recolha todos os objectos necessários na ilha, para que se consigam fazer duas poções mágicas para travar o que está a acontecer ao Duende.

Ao chegar à ilha Consonantis o herói encontra-se com cinco duendes, que o levam até ao interior de uma gruta e o jogador terá que apanhar alguns objectos de forma a completar missões. Por cada dois objectos que ele apanha recebe duas moedas. Após a colecta de todos os objectos o Herói aproxima-se do portão que o leva até á ilha Alfabeto, onde se encontra a Torre Mágica, que tem dois caldeirões à entrada. Ao chegar lá os cinco duendes indicam quais os objectos a colocar e em qual dos caldeirões, que os duendes levam depois até ao gigante para voltar ao seu estado normal.

Agora que o jogador sabe todo o alfabeto, consegue identificar quais as letras que faltam na entrada da Torre Mágica e sabe completar um último desafio. Com isto, o Feiticeiro desaparece numa nuvem de fumo e o herói ao entrar na sombria torre encontra o tesouro que fora roubado ao reino e ao abri-lo transforma o castelo num sítio cheio de cores e todas as nuvens escuras desaparecem.

3.1.2 Definição do Jogo

Com o *Wizard* é desejado que se consigam criar todos os níveis pertencentes ao jogo principal, e seguidamente são apresentados as características que os compõem.

Estes minijogos consistem num jogo em que o jogador controla um personagem animado que se move de um lado até ao outro do ecrã. Quando o jogador avança até ao limite do ecrã (da

direita para a esquerda), então a visão do jogo move-se também de forma a acompanhar o movimento do personagem. O personagem move-se na horizontal e está maior parte do tempo centrado no ecrã. O jogador tem uma visão lateral, a duas dimensões, de todo o cenário de jogo.

Nesta etapa o desafio apresentado ao jogador é o de colectar objectos flutuantes, denominados balões, que lhe permitam terminar com sucesso. Para que tal aconteça o jogador tem a hipótese de se mover para a esquerda, direita e saltar, conforme o *input*, através de teclado ou do rato. Este nível tem um número de balões existentes, uns com valor positivo, outros com valor negativo e ainda outros neutros. Toma-se como valor positivo o balão correcto, de acordo com a etapa em que se insere e os balões negativos os que não correspondem à dita característica. Por exemplo, numa etapa em que o jogador tem que recolher objectos que contenham a vogal “a”, então são positivos todos os balões que contenham essa vogal, em qualquer das suas formas, podendo ser em letra minúscula, maiúscula, manuscrita ou de “máquina”. Neste mesmo exemplo o jogador deve evitar apanhar um balão que contenha a letra “s”.

O jogador começa a etapa com uma barra de vida cheia, correspondente a três vidas, ou seja, ao terceiro erro que o jogador cometa então perde e tem que reiniciar o nível. Tendo em conta o exemplo anterior, se o jogador apanhar balões que contenham a letra “d”, “s” e “4”, então esvazia a sua barra de vida e terá que começar a etapa de novo.

Os balões neutros são aqueles que não influenciam directamente o resultado do jogo, sendo de captura opcional. Estes podem consistir em vidas extra, um aumento na velocidade do boneco, ou um escudo protector que evita que o jogador apanhe um balão negativo.

Durante todo o nível o personagem do jogador será ladeado por um ajudante, que o acompanha e que tem uma certa interacção com o jogador, alertando-o para certas situações. Pode avisar o jogador que apanhou o balão errado, ou que ganhou uma vida extra, ou que já está quase a acabar a etapa.

No decorrer da etapa existe uma música de fundo, um som ambiente específico do mapa que o jogador estiver a jogar. Também ocorrem sons esporádicos, igualmente relacionados com o ambiente em que o personagem se encontra. Por exemplo, se o cenário for uma floresta, existirá uma música ambiente definida para esse mapa e esporadicamente surgirá um som que se insira no tema, que poderá ser o cantar de pássaros ou o guinchar de macacos, entre outros.

3.2 Identificação e Caracterização do Problema

Com a criação de um *Wizard* pretende-se criar um ambiente, de fácil acesso e de interface amigável, que permita a um utilizador implementar um jogo, com base num modelo, sem a necessidade de haver uma grande equipa de programação, ou de ferramentas específicas bastante mais complexas, sendo suficiente a utilização de uma aplicação de fácil compreensão.

Dada a imensa variedade de tipos de videojogos existentes, foi considerado um jogo específico, criado pelo centro de desenvolvimento infantil Diferenças, para a implementação do *Wizard*. Foi escolhido este tipo de jogo para ser implementado o *Wizard* porque traria grandes benefícios ao referido centro poder contar com uma ferramenta deste género e assim criar diferentes jogos de acordo com as características dos alvos.

Conforme a definição do jogo efectuada, é determinado que o sistema a ser criado ilustre da melhor forma possível as características do jogo que foram identificadas, para que seja possível, através dos processos realizados, obter um produto final o mais próximo possível do desejado.

Como foi estudado no segundo capítulo, existem diversas ferramentas que facilitam a criação de jogos através de interfaces atraentes. Contudo, estas ferramentas, apesar de terem um objectivo facilitador, exigem que sejam manuseadas por utilizadores com conhecimentos de programação, de modelação de objectos, entre outros. É então que se encontra o grande obstáculo na criação de um jogo. No contexto de aplicação do *Wizard*, o centro Diferenças, há um grande querer na existência de videojogos que motivem crianças a entreterem-se com um jogo específico e que, inconscientemente, desenvolvem competências. Como o referido centro é especializado no acompanhamento de crianças “diferentes”, é seu desejo ter a possibilidade de adaptar o jogo a cada criança.

Com o intuito de um jogo adaptável e com o problema de desconhecimento de qualquer tecnologia de criação de jogos, então é desejado a criação de um sistema que ultrapasse o obstáculo ilustrado na Figura 3.3.

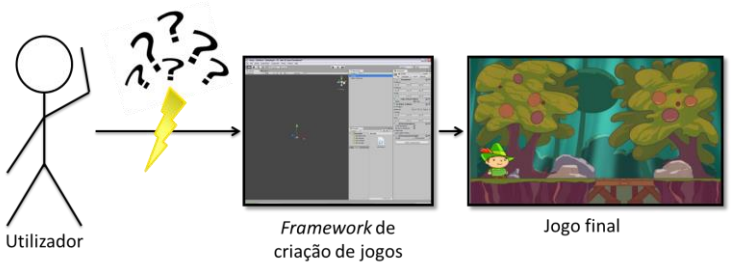

Figura 3.3 - Identificação do problema

3.2.1 Especificações técnicas

> *"Os requisitos de um sistema são fundamentalmente uma descrição das funções e restrições, que permitem a compreensão do problema do sistema que se pretende solucionar. Os requisitos podem guiar tanto o processo de desenvolvimento de software quanto o processo de aquisição." (Sommerville, Melnikoff, Arakaki, & de Andrade Barbosa, 2003)*

3.2.1.1 Requisitos Funcionais

Os requisitos funcionais são declarações de funções do sistema, da forma como o sistema reage a necessidades externas e comportamentos em determinadas situações. Descrevem assim as funcionalidades ou serviços que o sistema fornece. Dependem do tipo de sistema e do *software* utilizado na solução.

A referência RF significa Requisito Funcional e o número que se encontra após esta referência significa a sequência adoptada.

Tabela 3.1 - Requisitos Funcionais do sistema

RF	Descrição
RF001	Interface contendo todos os requisitos transmitidos na descrição do problema.
RF002	Interface deve ser operado essencialmente através do rato e do teclado.
RF003	Botões do interface deverão estar inactivos até haver uma acção de permissão.
RF004	Criação da base do sistema, como personagens, balões, etc.
RF005	O sistema deverá permitir ao utilizador a possibilidade de exportar todas as opções num determinado formato.
RF006	O sistema deve permitir ao utilizador importar componentes de opções tomadas anteriormente. Os componentes a serem importados devem estar no mesmo formato utilizado no RF005.

3.2.1.2 Requisitos não Funcionais

Requisitos não funcionais são aqueles que não dizem respeito directamente às funções específicas fornecidas pelo sistema (Sommerville et al., 2003). Podem estar relacionados a proprie-

dades e restrições do sistema aos processos de desenvolvimento. Ao contrário dos requisitos funcionais, não expressam qualquer função a ser realizada pelo sistema.

- **Operacionalidade:** O sistema irá operar em ambiente Windows. Deverá ser capaz de importar ficheiros Jpeg, Png, Wav, Obj, etc.

- **Confiabilidade:** No caso de alguma definição específica deixar de funcionar ou deixar de responder o sistema deverá ser capaz de manter um desempenho adequado, continuando as restantes etapas a funcionar com normalidade. Caso corra algum erro durante o carregamento de dados deverá ser carregado um ficheiro de *backup.*

- **Eficiência e Usabilidade:** O *software* deverá ser intuitivo, de modo que o utilizador saiba o que deve ser feito e como fazê-lo. O comportamento deverá ser estável e previsível ao utilizador.

- **Desempenho:** Embora não seja um requisito essencial ao sistema, deve ser considerada por corresponder a um factor de qualidade de *software.*

- **Padrão:** O interface deve ser padrão, assim como a linguagem de programação utilizada no desenvolvimento de *software*, para facilitar a manutenção e actualização do sistema. A utilização da *framework* que invoca o jogo também deve ser padrão.

3.3 Descrição da Solução

Considerando o modelo de jogo referido é possível descrever um conjunto de características imprescindíveis a serem incorporadas no *Wizard*.

O sistema a ser implementado terá que ter a capacidade de definir a dimensão do nível, o personagem principal e o seu ajudante, o som, os balões e todo o ambiente gráfico. Tem que ser utilizado um aspecto *user-friendly,*em que o utilizador saiba exactamente o deve fazer e qual será o resultado dessa acção, de modo a que consiga manusear o sistema a partir da primeira utilização. Para além disso a solução deve ser de rápido processamento e funcionar sobre o sistema operativo Windows.

O resultado das opções tomadas pelo utilizador deve ser exportado para um ficheiro que será importado para a *framework* responsável pela sua leitura e criação do jogo.

3.3.1 Processo de *Wizard*

Considerando o modelo de jogo descrito é apresentado o processo que o utilizador tem que efectuar de modo a definir o jogo desejado.

O primeiro passo necessário é definir uma etapa ou nível. A etapa é, basicamente, o conjunto de todos os objectos do nível, assim como de todas as interacções entre o jogador e os restantes elementos do jogo, ou seja, é considerado um minijogo dentro do jogo principal. No passo seguinte tem que ser definido o tema, relativamente aos aspectos sonoros. O passo seguinte é destinado a definir uma cena. Esta cena é composta por uma imagem de fundo e por um número de *targets*. Finalmente é necessário definir os balões que aparecem em cada cena. Todo este processo é posteriormente guardado num ficheiro de dados, como é demonstrado na Figura 3.4.

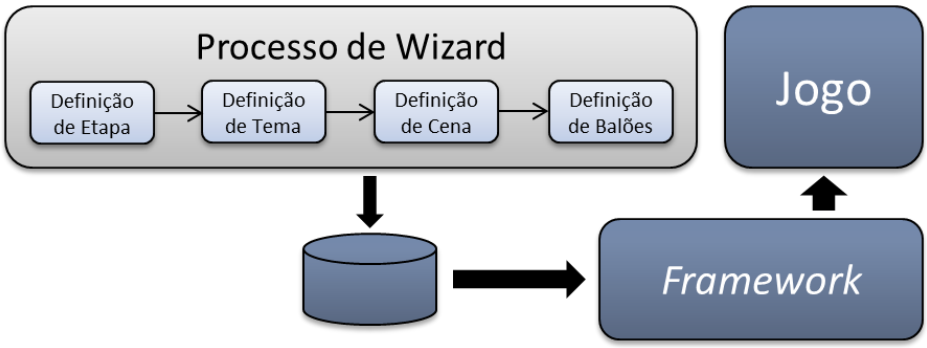

Figura 3.4 - Processo de *Wizard*

Uma etapa é composta por um conjunto de cenas, pelo seu nome, pelas instruções da etapa, pelo personagem principal e pelo seu ajudante. O elemento **nome da etapa** é o nome que este nível, ou missão terá, que poderá ser usada como referência em futuras instâncias. O número de cenas que a etapa tem é o que define o tamanho do nível, e como cada cena tem um número pré-definido de três balões fica implicitamente definido o número de balões. A etapa é composta por um conjunto de cenas que são apresentadas ao utilizador sequencialmente.

As **instruções** informam o jogador das regras que terá que cumprir, assim como dos objectivos do jogo, de modo a completar a missão com sucesso. A **personagem** é o elemento que é controlado pelo jogador e que é o protagonista herói do jogo. O **ajudante** é um personagem que não pode ser controlado pelo jogador, mas que o acompanha em toda a etapa, fornecendo dicas ou reparos dependendo se as acções do jogador foram acertadas ou não. Todo este processo é ilustrado na Figura 3.5.

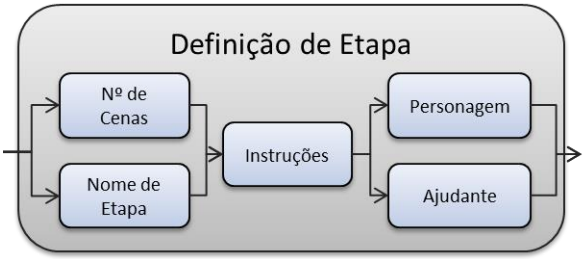

Figura 3.5 - Processo *Wizard* - Definição de Etapa

O processo de **definição de tema** não é mais do que a escolha do **som ambiente** do tema, como uma música que acompanha o jogador durante toda a etapa. O **som atómico** é um elemento sonoro que acontece esporadicamente no decorrer da missão, que deve ser um som breve e característico do ambiente escolhido.

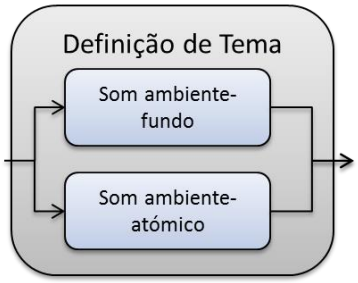

Figura 3.6 - Processo *Wizard* - Definição de Tema

A definição de cena consiste na escolha de uma **imagem de fundo**, que deverá ser concordante com o tema seleccionado. Para além disso, devem ser seleccionados os balões pertencentes a essa cena, podendo ser de três tipo diferentes. Os balões podem ser bons, maus ou neutros. Um **balão bom** é aquele que, de acordo com as instruções e o objectivo da etapa, o jogador terá que recolher no decorrer da missão de forma a completar a etapa com sucesso. Os **balões maus** são aqueles que o jogador não deve apanhar, pois são definidos como indesejados e fazem o jogador perder vidas.

Os **balões neutros** não fazem parte do conjunto de balões definidos como obrigatórios para o sucesso da missão, nem são definidos como maus, sendo assim de cariz opcional, tendo características especiais, podendo atribuir mais velocidade ao jogador, ou vidas extras.

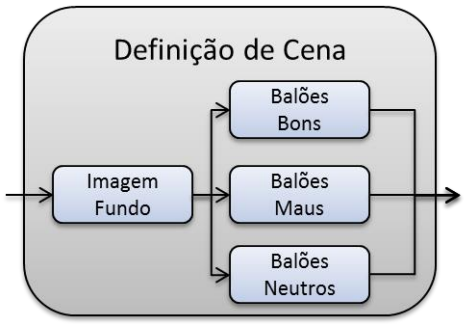

Figura 3.7 - Processo *Wizard* - Definição de Cena

Os balões são definidos atribuindo-lhes uma imagem própria, assim como sons característicos, de sucesso e de insucesso. O **som de sucesso** é ouvido quando o balão recolhido pertence ao objectivo da etapa e o **som de insucesso** soa quando o jogador apanha um balão que não faz parte desse objectivo.

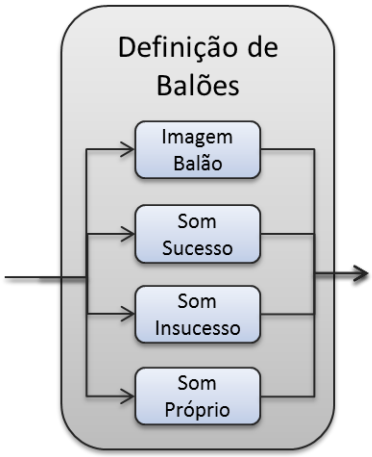

Figura 3.8 - Processo *Wizard* - Definição de Balões

Aquando da finalização deste processo, será criado um ficheiro armazenando toda a informação seleccionada, que possibilita a sua reutilização e após o seu término estará pronto para ser importado para o motor de jogo do Unity® que criará o jogo com todos os objectos seleccionados durante este processo.

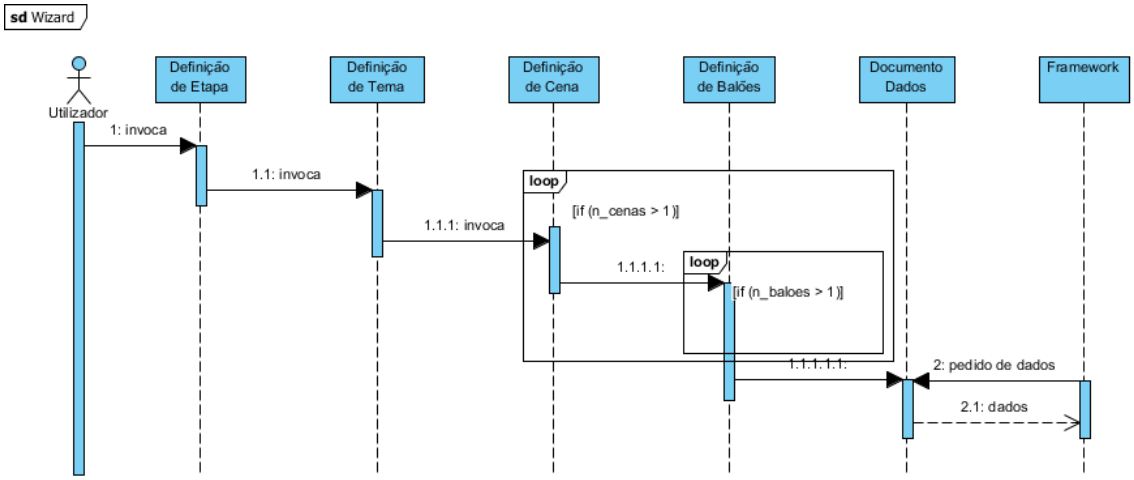

Figura 3.9 - Diagrama de Sequência do *Wizard*

A Figura 3.9 demonstra a sequência processual do *Wizard*. Ao executar o sistema o utilizador começa por definir a etapa, passando depois para a definição do tema. De seguida é necessário definir a cena e os seus balões. Como uma etapa pode ter mais que uma cena, então o seu processo repete-se um número de vezes igual ao número de cenas. De igual modo, uma cena poderá ter mais do que um balão e, se for esse o caso, o processo de definição de balões é repetido um número de vezes igual ao dos balões que a cena contém.

Terminado este processo é então criado um documento com todas as propriedades do jogo que foram definidas pelo utilizador, para posterior importação numa *framework* de edição.

4 Implementação do Protótipo

Os capítulos dois e três foram essenciais para a identificação de todos os requisitos fundamentais e assim produzir um sistema capaz de ser integrado num dos motores de jogo estudados, com o objectivo de criar um jogo.

4.1 Cenário de Utilização

O objectivo de um cenário é descrever como um utilizador deve explorar o sistema, essencialmente percebendo o comportamento do sistema.

Como se pode ver pela Figura 4.1, o utilizador opera o sistema através de um interface, que o guiará por todo o processo de definição do jogo.

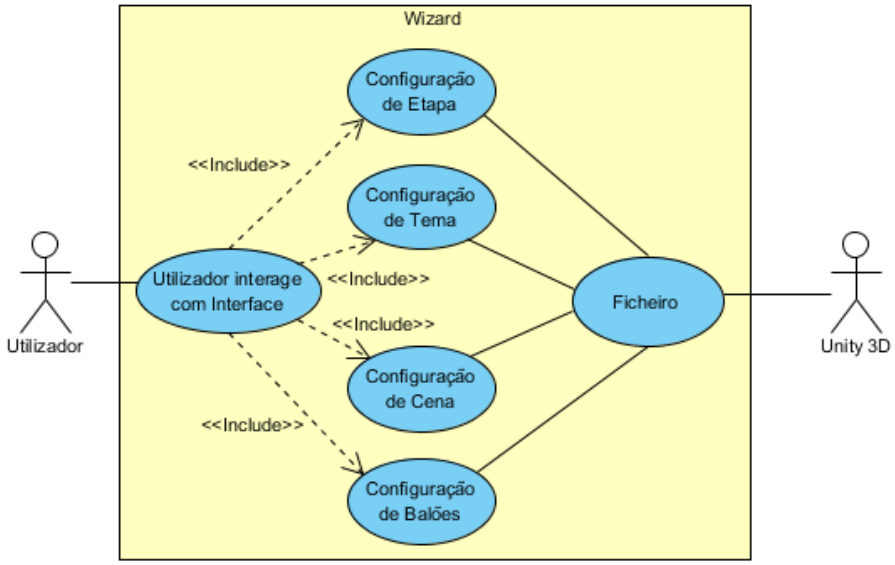

Figura 4.1 - Cenário de utilização do *Wizard*

4.2 Descrição das opções tomadas

Sendo que o propósito deste trabalho é a criação de um *software* que seja capaz de facilitar a criação de um jogo, a ferramenta seleccionada para a implementação da prova de conceito foi

o Unity®, pois a introdução ao ambiente de desenvolvimento é bastante amigável, de processos simples e é extremamente poderosa para utilizadores mais avançados.

Como se pode ver pela Tabela 2.2, foi feito um levantamento das características principais de cada um dos motores de jogo estudados e foi atribuída uma classificação de acordo com as suas particularidades e com os requisitos necessários para a concepção do protótipo.

O Unity® foi a ferramenta escolhida em detrimento de outras pela sua facilidade de utilização, pela possibilidade de publicar jogos multiplataforma, sendo fácil a exportação para *tablets* e para *web browsers*. O tamanho da sua comunidade e a imensa quantidade de recursos existentes *online* foi um factor importante nesta escolha, assim como o facto da sua linguagem de programação ser C# e JavaScript, que são largamente utilizadas.

O jogo de computador Reino dos Fonemas, como foi descrito, é composto por várias aventuras diferentes, que são considerados minijogos dentro do jogo principal. Para a prova e validação do conceito de *Wizard* foi apenas considerado um desses minijogos, sendo criado um modelo desse modo de jogo.

O minijogo escolhido faz parte do conjunto de etapas da ilha “Vogalis”, que é considerado o que apresenta um maior grau de importância. Optou-se então por desenvolver o *Wizard* para que, através dele, se possam construir várias e diferentes etapas que serão integradas no jogo principal.

4.3 Implementação

Face aos conceitos descritos anteriormente, o *Wizard* é constituído por diversas interfaces que permitem o utilizador seguir todos os passos para a criação do jogo. A sequência de etapas é apresentada na Figura 4.2, que serão depois detalhadas individualmente.

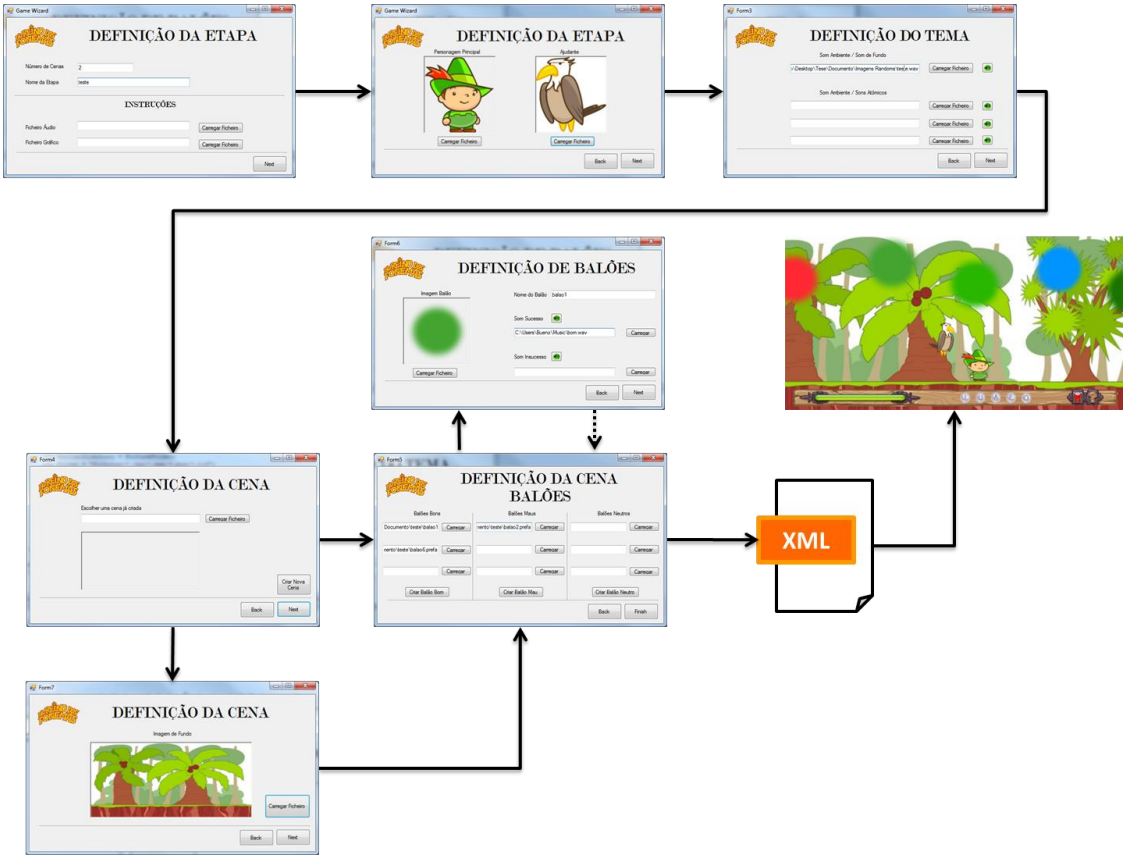

Figura 4.2 - Interface das várias etapas do *Wizard*

Em primeiro lugar, surge ao utilizador um ecrã, Figura 4.3, no qual tem que inserir quantas cenas deverão existir numa etapa, qual o nome da etapa e quais são os ficheiros áudio e gráficos com as instruções relativas à etapa, assim como o objectivo do jogador no decorrer da missão.

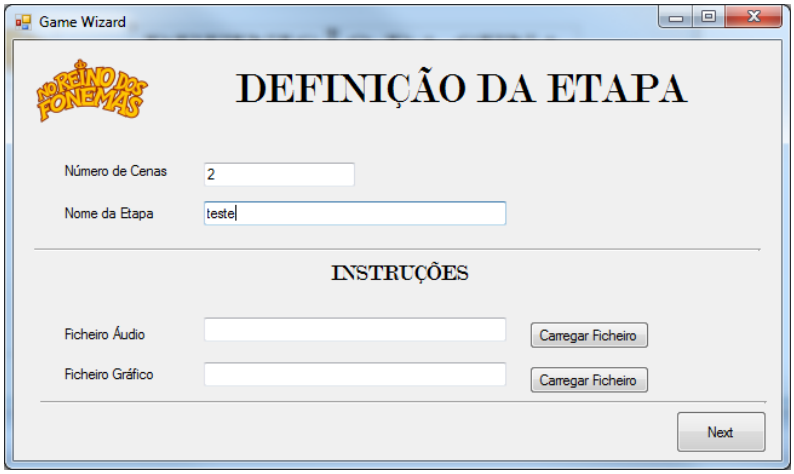

Figura 4.3 - *Wizard* - Definição de Etapa - Ecrã 1

Após a selecção destes aspectos mais globais do jogo, é carregada a interface seguinte, Figura 4.4, sugerindo ao utilizador que escolha qual a personagem principal do jogo e o seu ajudante. É pedido ao utilizador que clique na área correspondente à personagem do jogador e é solicitado que o utilizador indique qual é o ficheiro com a imagem do personagem desejado, sendo este mostrado na mesma área que foi clicada de início. Para atribuição de um ficheiro gráfico ao ajudante o processo é em tudo idêntico ao da personagem principal, apenas com a diferença que este tem uma área de clique e de visualização própria.

Figura 4.4 - *Wizard* - Definição de Etapa - Ecrã 2

Após terem sido escolhidos o jogador e o seu ajudante o utilizador clica no botão seguinte (*Next*), que o leva ao ecrã de definição do tema, mostrado na Figura 4.5. Nesta interface é solicitado ao utilizador que seleccione o ficheiro áudio do som ambiente, que se ouvirá durante o decorrer de toda a etapa, assim como dá a possibilidade de serem seleccionados três sons ambiente atómicos, que soarão aleatoriamente no decorrer da etapa.

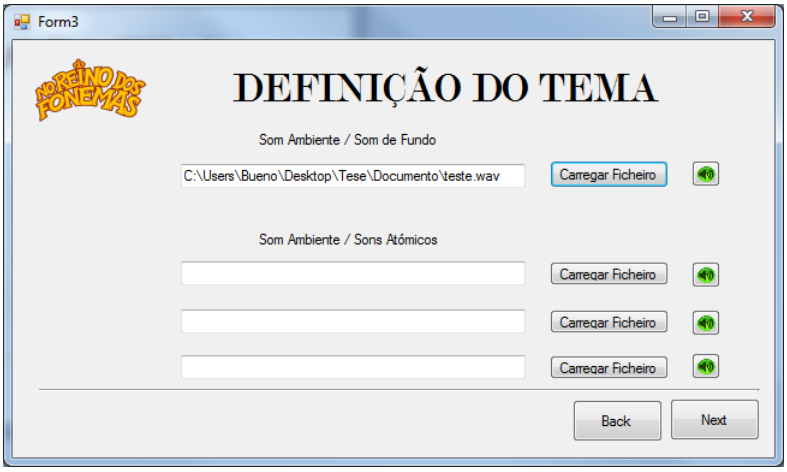

Figura 4.5 - *Wizard* - Definição de Tema - Ecrã 1

Ao avançar o utilizador é guiado até ao painel de definição de cenas, Figura 4.6. Aqui, o utilizador pode escolher uma cena já criada anteriormente, tendo um campo próprio para esse efeito, e que mostra uma pré-visualização numa caixa mais abaixo, clicando depois no botão *Next*, que o levará ao ecrã 2 da definição de cenas. Existe também um botão em que o utilizador pode clicar, que o leva ao painel de criação de cenas, caso seja necessário. Isto leva-o ao ecrã 1.1 da definição de cenas, mostrado na Figura 4.7.

Em ambos os casos, independentemente do utilizador ter escolhido uma cena já criada ou ter definido uma nova, ao clicar no botão *Next* é conduzido ao ecrã 2 da definição de cenas, Figura 4.8.

Esta etapa de criação de cenas repete-se pelo número de cenas que o utilizador deseja, que já foi definido no ecrã 1 da definição de etapa, que se encontra acima, na Figura 4.3.

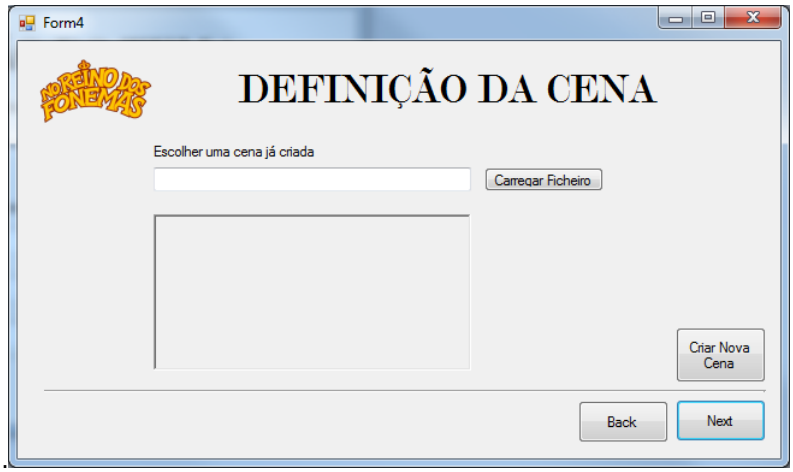

Figura 4.6 - *Wizard* - Definição de Cena - Ecrã 1

Se o utilizador tiver clicado no botão de **criar uma nova cena** será levado ao ecrã1.1 da definição de etapa, mostrado na Figura 4.7, onde é solicitado para clicar numa determinada área, que permite seleccionar uma imagem de fundo para a sua cena, que será depois mostrada nesse mesmo espaço. Após o clique no botão *Next* será levado até ao passo seguinte.

Figura 4.7 - *Wizard* - Definição de Cena - Ecrã 1.1

Qualquer que tenha sido a opção que o utilizador tomou anteriormente, este é guiado até ao ecrã que está apresentado na Figura 4.8.

Caso o utilizador tenha seleccionado uma cena já criada no ecrã1 da definição de cena então este painel será preenchido com os balões pertencentes à cena seleccionada, que são carregados do ficheiro de armazenamento e apresentado nos devidos campos de balões bons, maus e neutros. Por outro lado, se o utilizador criar uma nova cena, então os campos mostrados na Figura 4.8 não estarão preenchidos.

O utilizador pode retirar ou alterar um balão que já esteja seleccionado, independentemente do seu tipo, assim como adicionar balões já criados e que estão guardados num ficheiro. Pode também optar por criar novos balões de qualquer um dos diferentes tipos, clicando num dos botões, que o levam até ao painel da Figura 4.9.

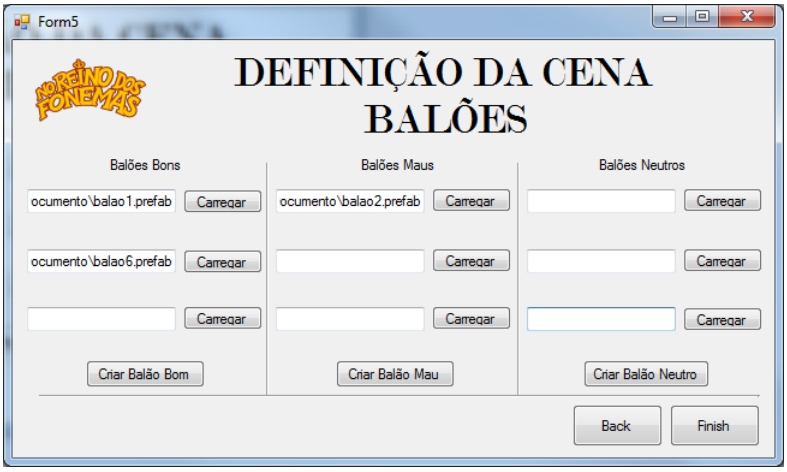

Figura 4.8 - *Wizard* - Definição de Cena - Ecrã 2

Quando o utilizador deseja criar um novo balão é guiado até ao interface mostrado na Figura 4.9, em que é requisitado que clique na área definida para seleccionar a imagem do balão, que será depois mostrado nesse mesmo espaço e é também necessário que o utilizador escolha os sons de sucesso e de insucesso alusivos a esse balão. Caso seja um balão neutro, os sons devem ser iguais.

Depois de criado o balão o utilizador carrega no botão *Next* e é levado até ao painel anterior, que é o ecrã 2 da definição de cenas.

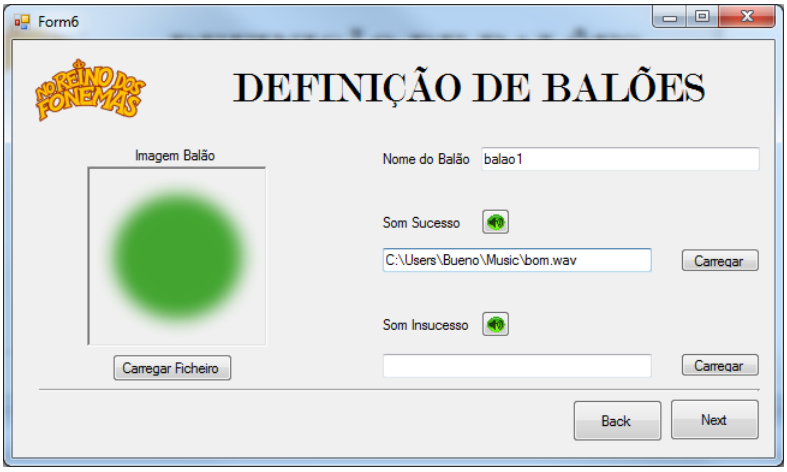

Figura 4.9 - *Wizard* - Definição de Balões - Ecrã 1

Quando estiveram todos os botões desejados seleccionados o utilizador clica no botão *Finish* e o *Wizard* é terminado, com os objectos inseridos pelo utilizador a ficarem gravados num ficheiro XML com uma estrutura semelhante à que está apresentada na Figura 4.10.

```
<?xml version="1.0"?>
<etapa>
        <nome></nome>
        <n_cenas></n_cenas>
        <instrucoes>
                <inst_audio></inst_audio>
                <inst_graf></inst_graf>
        </instrucoes>
        <personag></personag>
        <ajudante></ajudante>

        <tema>
                <amb_fundo></amb_fundo>
                <amb_atom></amb_atom>
        </tema>

        <cena>
                <cena_n></cena_n>
                <img_fundo></img_fundo>

                <bal_bom1></bal_bom1>
                <bal_bom2></bal_bom2>
                <bal_bom3></bal_bom3>

                <bal_mau1></bal_mau1>
                <bal_mau2></bal_mau2>
                <bal_mau3></bal_mau3>

                <bal_neut1></bal_neut1>
                <bal_neut2></bal_neut2>
                <bal_neut3></bal_neut3>
        </cena>

        <balao>
                <nome_bal></nome_bal>
                <img_bal></img_bal>
                <som_suc></som_suc>
                <som_insuc></som_insuc>
        </balao>
</etapa>
```

Figura 4.10 - *Wizard* - Estrutura de *Output*

4.4 Validação

Após a implementação da plataforma descrita, foi possível efectuar um conjunto de testes para avaliar toda a aplicação, com o intuito de melhorar e tentar perceber possíveis características a desenvolver. Os testes efectuados consistem na execução da aplicação *Wizard* e na avaliação do resultado obtido, que é mostrado na Figura 4.11, que foi depois importado para a ferramenta Unity, que criou o ambiente de jogo apresentado na Figura 4.12.

Para testar o resultado da solução foram seguidos todos os passos da Figura 4.2, e inseridos componentes em alguns campos, como é mostrado da Figura 4.3 até à Figura 4.9. Propositadamente foram deixados em branco alguns campos, que segundo os requisitos funcionais do sistema, não deveriam ser obrigatórios para o bom funcionamento do jogo. Caso disso é a falta de alguns ficheiros de som, que embora criem alguma imersão no ambiente de jogo, não deverão ser considerados elementos críticos para a boa execução do jogo.

Assim, de acordo com o processo do *Wizard* seguido, é esperado que seja criado um ficheiro XML com a estrutura apresentada na Figura 4.10, com "teste" como nome da etapa, com duas cenas e sem ficheiros de instruções. É esperado que o personagem principal seja uma figura de nome "berto" e que o seu ajudante tenha o nome de "aguia1". É também esperado que o nome do ficheiro de som ambiente seja "teste".

Definidos os aspectos referentes à configuração global da etapa é então definida cada uma das cenas. A primeira cena deverá ter uma imagem de fundo com o nome "img_fundo1" e deverá conter três balões, dois bons e um mau. Os balões bons deverão ser os "balao1" e "balao6" e o balão mau deverá ser o "balao2".

A segunda cena terá a imagem de fundo com o nome de "img_fundo2", os balões bons com os nomes "balao6" e "balao1" e o balão neutro com nome "balao3".

Como se pode ver na Figura 4.9, foi definido o "balao1" com uma imagem, com o seu nome, e com o som de sucesso de "bom". Este processo foi repetido para os restantes balões, com os respectivos nomes e com imagens diferentes para cada balão e deixando os seus campos referentes aos sons vazios.

A Figura 4.11 apresenta o ficheiro XML que é obtido através da execução da aplicação *Wizard* e apresenta a estrutura esperada e com os campos referentes a cada elemento preenchidos da forma desejada.

```
<?xml version="1.0"?>
<etapa>
        <nome>teste</nome>
        <n_cenas>2</n_cenas>
        <instrucoes>
                <inst_audio></inst_audio>
                <inst_graf></inst_graf>
        </instrucoes>
        <personag>berto.jpg</personag>
        <ajudante>aguia1.jpg</ajudante>

        <tema>
                <amb_fundo>teste.wav</amb_fundo>
                <amb_atom></amb_atom>
        </tema>

        <cena>
                <cena_n>1</cena_n>
                <img_fundo>img_fundo1.jpg</img_fundo>

                <bal_bom1>balao1</bal_bom1>
                <bal_bom2>balao6</bal_bom2>
                <bal_bom3></bal_bom3>

                <bal_mau1>balao2</bal_mau1>
                <bal_mau2></bal_mau2>
                <bal_mau3></bal_mau3>

                <bal_neut1></bal_neut1>
                <bal_neut2></bal_neut2>
                <bal_neut3></bal_neut3>
        </cena>
        <cena>
                <cena_n>2</cena_n>
                <img_fundo>img_fundo2.jpg</img_fundo>

                <bal_bom1>balao6</bal_bom1>
                <bal_bom2>balao1</bal_bom2>
                <bal_bom3></bal_bom3>

                <bal_mau1></bal_mau1>
                <bal_mau2></bal_mau2>
                <bal_mau3></bal_mau3>

                <bal_neut1>balao3</bal_neut1>
                <bal_neut2></bal_neut2>
                <bal_neut3></bal_neut3>
        </cena>

        <balao>
                <nome_bal>balao1</nome_bal>
                <img_bal>balao1.jpg</img_bal>
                <som_suc>bom.wav</som_suc>
                <som_insuc></som_insuc>
        </balao>
        <balao>
                <nome_bal>balao2</nome_bal>
                <img_bal>balao2.jpg</img_bal>
                <som_suc></som_suc>
                <som_insuc></som_insuc>
        </balao>
        <balao>
                <nome_bal>balao3</nome_bal>
                <img_bal>balao3.jpg</img_bal>
                <som_suc></som_suc>
                <som_insuc></som_insuc>
        </balao>
        <balao>
                <nome_bal>balao6</nome_bal>
                <img_bal>balao6.jpg</img_bal>
                <som_suc></som_suc>
                <som_insuc></som_insuc>
        </balao>
</etapa>
```

Figura 4.11 - *Output* de um teste do *Wizard*

Foi depois importado o ficheiro XML obtido para a *framework* Unity®, de modo a criar um jogo com os componentes e as características definidas. O resultado da execução do jogo, em ambiente Unity® e importando o ficheiro de *output* do *Wizard*, é o mostrado na Figura 4.12.

O resultado é um jogo totalmente funcional, com o herói e o ajudante correspondente ao que foi definido, assim como as imagens de fundo e os balões.

Na mesma imagem denota-se uma ligeira fractura do cenário, na base da imagem, entre as vogais “e” e “o”, derivado à junção das duas cenas. É então possível confirmar a presença de dois balões bons (verdes) e um mau (vermelho) na primeira cena, e de dois balões bons (verdes) e um neutro (azul) na segunda cena, como era esperado.

Figura 4.12 - *Printscreen* do jogo resultante

A utilização do *Wizard* comprova que é possível a existência de uma ferramenta que auxilie um utilizador na criação de jogos.

Relacionando a questão principal deste trabalho, então são apresentadas evidências que a utilização do *Wizard* permite que um utilizador sem conhecimentos de programação crie um jogo, desde que seja aplicado a este modelo.

5 Conclusões

Neste capítulo é feito um sumário desta dissertação, evidenciando os aspectos mais importantes deste trabalho, assim como uma potencial direcção para futuros trabalhos.

5.1 Síntese

A solução apresentada funciona como apoio ao processo de criação de jogos electrónicos, quebrando o obstáculo que um utilizador comum encontra ao deparar-se com um ambiente de programação de jogos. Este sistema, o *Wizard*, facilita o utilizador no processo de criação de um jogo electrónico, sem ser preciso dominar vários aspectos necessários para a construção de um videojogo, nem é obrigatória a existência de uma grande equipa, com elementos especializados em diferentes áreas. Esta abordagem foi alcançada com uma aplicação integrada sobre um motor de jogo, em que o utilizador segue uma série de passos definidos para a sua criação.

O projecto abrange todas as características obrigatórias de uma interface simples e intuitiva, assim como todos os atributos necessários à constituição do jogo, de forma a simplificar a sua criação.

No contexto educacional em que foi pensada, esta solução tem grandes vantagens para o centro Diferenças, pois permite-lhes a personalização de videojogos conforme as necessidades da criança alvo. A necessidade destes jogos é justificada pelo aumento motivacional que criam nos alunos, estimulando-os com diferentes oportunidades de resolução de problemas. Estes problemas são oferecidos pelos jogos educativos e proporcionam um ambiente crítico mais cativante do que o ensino tradicional, mantendo o objectivo de desenvolverem capacidades cognitivas nas crianças.

Atendendo ao facto que **Reino dos Fonemas** é um jogo bastante extenso e com etapas variadas, este trabalho foca-se na implementação de um tipo pré-definido de uma dessas etapas, sendo possível a variação de atributos de jogo, mas nunca a sua mecânica. Contudo, o objectivo de possibilitar a criação de um jogo com um sistema auxiliar, na lógica *Wizard*, foi conseguido.

5.2 Trabalho Futuro

Um trabalho de melhoramento que poderia ser aplicado à solução proposta é a possibilidade de um *Wizard* para a criação do jogo completo. Esta solução abrange apenas um dos minijogos presentes no jogo Reino dos Fonemas, então uma abordagem interessante seria a criação de um modelo do jogo completo, de maneira a que um *Wizard* que abranja todos os modos de jogo integrados sirva como suporte à criação do jogo total.

Poderia também ser vantajosa uma abordagem para a criação da mecânica do jogo, dos seus objectivos, dos seus desafios e das acções dos personagens, criando assim um modelo de jogo totalmente diferente do apresentado, visto que neste projecto não é possível definir a mecânica do jogo, que já está montada, servindo como modelo.

6 Bibliografia

Abt, C. C. (1974). *Jogos simulados: estratégias e tomada de decisão*: Jose Olympio.

Anderson, E. F., Engel, S., Comninos, P., & McLoughlin, L. (2008). *The case for research in game engine architecture.* Paper presented at the Proceedings of the 2008 Conference on Future Play: Research, Play, Share.

Apperley, T. (2010). What games studies can teach us about videogames in the English and Literacy classroom. *Australian Journal of Language and Literacy, The, 33*(1), 12.

Bailey, M., & Games, Y. (2011). Using Game Maker.

Busby, J., Parrish, Z., & Wilson, J. (2009). *Mastering Unreal Technology, Volume I: Introduction to Level Design with Unreal Engine 3* (Vol. 1): Pearson Education.

Cabral, F. (1997). Jogos Eletrônicos: Técnica Ilusionista ou Emancipadora? *Revista USP*(35).

Cardoso, T. (2013). Aula de Apresentação *Tecnologia de Jogos Digitais*: FCT - UNL.

Chan, T. S., & Ahern, T. C. (1999). Targeting motivation-adapting flow theory to instructional design. *Journal of Educational computing research, 21*(2), 151-164.

Creighton, R. H. (2010). *Unity 3D Game Development by Example: A Seat-of-Your-Pants Manual for Building Fun, Groovy Little Games Quickly*: Packt Publishing Ltd.

Demaine, E. D., Hohenberger, S., & Liben-Nowell, D. (2003). Tetris is hard, even to approximate *Computing and Combinatorics* (pp. 351-363): Springer

Diferenças.). Centro de Desenvolvimento Infantil. from www.diferencas.net

Elliott, J. L. (2013). *HTML5 Game Development with GameMaker*: Packt Publishing Ltd.

Flausino, R. (2006). Os Jogos Eletrônicos e seus Impactos na Sociedade. from http://gamehall.uol.com.br/selectgame/os-jogos-eletronicos-e-seus-impactos-na-sociedade/

Friedman, T. (1999). The semiotics of SimCity. *First Monday, 4*(4).

Gallagher, S. (2012). Math Blaster *Review*. from http://www.avatargeneration.com/2012/08/math-blaster-review/

Goto, M. R. (2005). Evoluindo a diversão. *EGM Brasil, 35,* 46-55.

Greenfield, P. M., Camaioni, L., Ercolani, P., Weiss, L., Lauber, B. A., & Perucchini, P. (1994). Cognitive socialization by computer games in two cultures: Inductive discovery or mastery of an iconic code? *Journal of Applied Developmental Psychology, 15*(1), 59-85.

Gregory, J. (2009). *Game engine architecture*: CRC Press.

Gros, B. (2007). Digital games in education: The design of games-based learning environments. *Journal of Research on Technology in Education, 40*(1), 23-38.

Johnson, S. (2006). Tudo o que é mau faz bem: como os jogos de vídeo, a TV ea Internet nos estão a tornar mais inteligentes. *Lisboa: Lua de Papel*.

Joseph, E. (2014). Bot Colony – a Video Game Featuring Intelligent Language-Based Interaction with the Characters.

Kickmeier-Rust, M. D., Peirce, N., Conlan, O., Schwarz, D., Verpoorten, D., & Albert, D. (2007). Immersive digital games: the interfaces for next-generation e-learning? *Universal Access in Human-Computer Interaction. Applications and Services* (pp. 647-656): Springer

Koster, R. (2013). *Theory of fun for game design*: " O'Reilly Media, Inc.".

Lane, M. (2007). Making History: The Calm & the Storm. *Strategy First, Montreal, Canada*.

McFarlane, A., Sparrowhawk, A., & Heald, Y. (2002). Report on Educational Use of Games: Teachers Evaluating Educational Multimedia Report: TEEM, Retrieved.

Moratori, P. B. (2003). Por que utilizar jogos educativos no processo de ensino aprendizagem. *UFRJ. Rio de Janeiro*.

Muñoz Rueda, A. (2012). 3D Home: Una interfaz para el control de un hogar inteligente.

Newman, J. (2013). *Videogames*: Routledge.

Nilson, B., & Söderberg, M. (2007). Game Engine Architecture.

Nussbaum, M., Rosas, R., Rodríguez, P., Sun, Y., & Valdivia, V. (1999). Diseño, desarrollo y evaluación de videojuegos portátiles educativos y autorregulados. *Ciencia al día, 3*(2), 1-20.

Overmars, M. (2009). Designing games with game maker: Version.

Owens, T. (2011). Modding the history of science: Values at play in modder discussions of Sid Meier's Civilization. *Simulation & Gaming, 42*(4), 481-495.

Rapeepisarn, K., Wong, K. W., Fung, C. C., & Khine, M. S. (2008). The relationship between game genres, learning techniques and learning styles in educational computer games *Technologies for E-Learning and Digital Entertainment* (pp. 497-508): Springer

Riedl, M. O. (2010). Scalable personalization of interactive experiences through creative automation. *Computers in Entertainment (CIE), 8*(4), 26.

Rodrigo, M. M. T. (2010). Dynamics of student cognitive-affective transitions during a mathematics game. *Simulation & Gaming*.

Schafersman, S. (1994). An introduction to science: Scientific thinking and the scientific method. *online whitepaper, January*.

Sherrod, A. (2006). *Ultimate 3D Game Engine Design & Architecture*: Charles River Media, Inc.

Sommerville, I., Melnikoff, S. S. S., Arakaki, R., & de Andrade Barbosa, E. (2003). *Engenharia de software* (Vol. 6): Addison Wesley.

Squire, K. (2005). *Game-based learning: Present and future state of the field*: Masie Center e-Learning Consortium.

Tavares, R. (2013). Games na Educação: A Batalha está começando.

Trenholme, D., & Smith, S. P. (2008). Computer game engines for developing first-person virtual environments. *Virtual reality, 12*(3), 181-187.

Viana, T. C. (2009). Projeto e desenvolvimento de um jogo de futebol utilizando o motor de jogo Unity 3D.

Wiki, M. B.). About the Blaster Learning System. from http://mathblaster.wikia.com/wiki/Math_Blaster_Wiki